LaTeX2Nemeth and the amsmath package

Andreas Papasalouros, Antonis Tsolomitis

1 Introduction

LaTeX2Nemeth is a software package written in Java that converts LaTeX files written in the UTF-8 encoding to Braille using the Nemeth standard for mathematics. It has supported more than 800 mathematics symbols up to now and a great amount of mathematics structures from simple exponents to multiline equation arrays.

Recently the project attracted the attention of the TeX development fund, and additional support was given to cover the American Mathematical Society (AMS) packages and extended Unicode mathematics support, including all symbols found in `unimath-symbols.pdf`, to the extent that this is possible by the structure of the Nemeth standard.

The current version of latex2nemeth found on CTAN incorporates these additions. If something fails to work and is neither described here nor is a capability of the AMS packages that has only a visual effect (and hence irrelevant to the blind), it should be considered a bug, and it should be reported to the authors as such.

As for languages, it supports the Latin alphabet (so English is supported in grade1 Braille) and it also supports the Greek language, both monotonic and polytonic.

2 Extended Unicode mathematics support

All symbols in Will Robertson's `unimath-symbols.pdf` file are supported with the exception of the symbols found in the table at the end of the article. We found no way to support those in the Nemeth standard. If someone knows how to support them we will gladly add them. Please contact us in such a case.

The unsupported symbols listed at the end of the article (in Section 13) are only 95 out of the 2441 symbols found in `unimath-symbols.pdf`. So more than 96% of the symbols are supported. In practice the number is even higher, since out of those unsupported symbols, the ones at the beginning of the table that are used to compose large operators or delimiters are either irrelevant to the blind (such as the pieces of the parenthesis) or both irrelevant to the blind and unsupported by `xelatex/lualatex`, such as the pieces that compose large integrals or sums.

3 Support for the `amsmath` package

Most structures of the `amsmath` package are supported. Unsupported features are those irrelevant to the blind (things that have only visual interest), for example options that set the placement of tags (e.g., centertags), and the commands provided by the `amscd` and `amsxtra` packages.

3.1 Displayed equations

All of the environments below are supported:

```
equation   equation*   align     align*
gather     gather*     alignat   alignat*
multline   multline*   flalign   flalign*
split
```

Let us take an example from the AMS documentation. The code

$$a_1 = b_1 + c_1 \tag{1}$$
$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{2}$$

will produce

⠁⠼⠂⠐ ⠨⠅ ⠃⠼⠂⠐⠬⠉⠼⠂⠐ ⠷⠼⠂⠾
⠁⠼⠆⠐ ⠨⠅ ⠃⠼⠆⠐⠬⠉⠼⠆⠐⠤⠙⠼⠆⠐⠬⠑⠼⠆⠐ ⠷⠼⠆⠾

Notice that latex2nemeth will always use the number indicator ⠼ before a number even if the Nemeth standard allows its absence in some cases, as in indices, to save space. Also notice

doi.org/10.47397/tb/43-3/tb135papasalouros-amsmath

that commands that modify the spacing (like `\multlinegap`) are not supported as they are of no use for the blind and embossers.

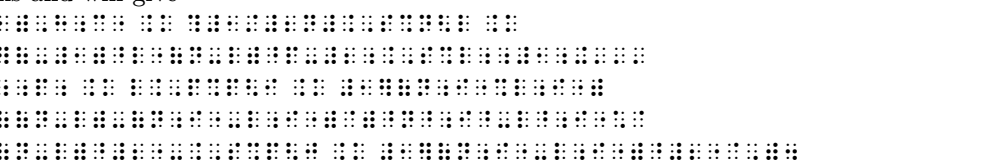
Another example taken from the amsmath documentation is

$$H_c = \frac{1}{2n}\sum_{l=0}^{n}(-1)^l(n-l)^{p-2}\sum_{l_1+\cdots+l_p=l}\prod_{i=1}^{p}\binom{n_i}{l_i} \cdot [(n-l)-(n_i-l_i)]^{n_i-l_i}\cdot\Big[(n-l)^2-\sum_{j=1}^{p}(n_i-l_i)^2\Big]. \tag{3}$$

with code

```
\begin{equation}\label{e:barwq}\begin{split}
H_c&=\frac{1}{2n} \sum^n_{l=0}(-1)^{l}(n-{l})^{p-2}
  \sum_{l _1+\dots+ l _p=l}\prod^p_{i=1} \binom{n_i}{l _i}\\
&\quad\cdot[(n-l )-(n_i-l _i)]^{n_i-l _i}\cdot
  \Bigl[(n-l )^2-\sum^p_{j=1}(n_i-l _i)^2\Bigr].
\end{split}\end{equation}
```

works and will give



where we added some line breaks by hand to help typeset the Braille dots for this article.

The "-ed" environments such as `aligned` and `cases` etc are also supported (notice that `\(` and `\)` are supported as well as single dollar signs, and `\[` and `\]` as well as double dollars):

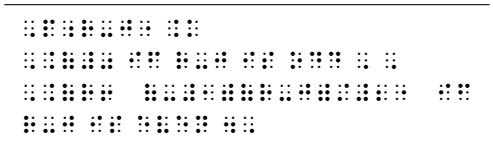
Code

```
$$ P_{r-j}=\begin{cases}
  0& \text{if $r-j$ is odd},\\
r!\,(-1)^{(r-j)/2}&
  \text{if $r-j$ is even}.
\end{cases}
$$
```

TEX

$$P_{r-j} = \begin{cases} 0 & \text{if } r-j \text{ is odd},\\ r!\,(-1)^{(r-j)/2} & \text{if } r-j \text{ is even}. \end{cases}$$

Braille



3.2 Display interruption

`\intertext` (as well as `\shortintertext`, from mathtools) is supported:

Code:

```
\begin{align}
A_1&=N_0(\lambda;\Omega')-\phi(\lambda;\Omega'),\\
A_2&=\phi(\lambda;\Omega')-\phi(\lambda;\Omega),\\
\intertext{and}
A_3&=\mathcal{N}(\lambda;\omega).
\end{align}
```

TEX:

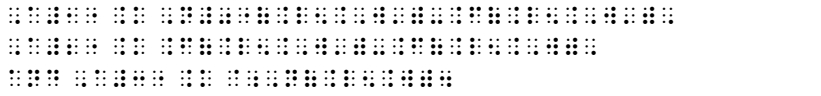
$$A_1 = N_0(\lambda;\Omega') - \phi(\lambda;\Omega'), \tag{4}$$

$$A_2 = \phi(\lambda;\Omega') - \phi(\lambda;\Omega), \tag{5}$$

and

$$A_3 = \mathcal{N}(\lambda;\omega). \tag{6}$$

Braille:



3.3 Equation numbering

Equation numbering with the standard `\label` and `\ref` mechanism is supported as well as with `\eqref`. Commands that modify the style of the references (such as a change of fonts) make no sense for the blind and are not supported.

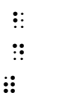
4 Miscellaneous mathematical features

4.1 Matrices

Matrix environments as well as `\hdotsfor` are supported

Code	TEX output	Braille
`$$ \begin{matrix} a&b&c&d\\ e&\hdotsfor{3} \end{matrix}$$`	$a \quad b \quad c \quad d$ $e \quad \ldots\ldots$	

Small matrices (`smallmatrix`) too:

Code	TEX output	Braille
`$\bigl( \begin{smallmatrix} a&b\\ c&d \end{smallmatrix} \bigr)$`	$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$	

Parenthesized matrices (`pmatrix`), as well as `bmatrix` and `Bmatrix` too:

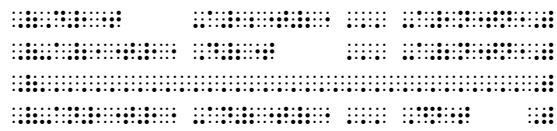
Code

```
$$\begin{pmatrix}
D_1t&-a_{12}t_2&\dots&-a_{1n}t_n\\
-a_{21}t_1&D_2t&\dots&-a_{2n}t_n\\
\hdotsfor[2]{4}\\
-a_{n1}t_1&-a_{n2}t_2&\dots&D_nt
\end{pmatrix}$$
```

TEX output

$$\begin{pmatrix} D_1t & -a_{12}t_2 & \dots & -a_{1n}t_n \\ -a_{21}t_1 & D_2t & \dots & -a_{2n}t_n \\ \hdotsfor{4} \\ -a_{n1}t_1 & -a_{n2}t_2 & \dots & D_nt \end{pmatrix}$$

Braille



4.2 Math spacing commands

Math spacing commands are irrelevant to the blind so they are ignored. However, a warning will be printed on standard output about unknown math symbols.

4.3 Dots

All dot commands `\dotsc`, `\dotsb`, `\dotsm`, `\dotsi`, and `\dotso` are supported.

4.4 Nonbreaking dashes

Nonbreaking dashes are irrelevant to the blind so they are ignored.

4.5 Accents in math

All accents are supported but commands related to better positioning of the accents are irrelevant to the blind.

$\hat{\hat{A}}$ is `$\hat{\hat{A}}$` and gives ⠠⠁⠣⠸⠣⠣⠸⠣

4.6 Roots

Any kind of root is supported but `\leftroot` and `\uproot` are irrelevant to the blind and ignored.

$\sqrt[\beta]{k}$ is `$\sqrt[\leftroot{-2}\uproot{2}\beta]{k}$` and gives ⠣⠨⠃⠜⠅⠻

4.7 Boxed formulas

Boxes around formulas are irrelevant to the blind and ignored. However, the contents of `\boxed` will be transcribed provided that the whole `\boxed` command is inside math mode. So `$\boxed{x=1}$` will work, but `\boxed{x=1}` will fail (although LaTeX works with both).

4.8 Over and under arrows

All over and under arrows are supported. For example,

$\underleftarrow{x}$ is `$\underleftarrow{x}$` and gives ⠭⠩⠫⠪⠒⠒⠻

4.9 Extensible arrows

`\xleftarrow` and `\xrightarrow` are supported:

$A\xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1}C$ is `$A\xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1}C$` and gives ⠠⠁⠫⠪⠒⠒⠣⠝⠬⠨⠍⠤⠼⠂⠻⠠⠃⠫⠒⠒⠕⠩⠠⠞⠣⠝⠬⠤⠊⠤⠼⠂⠻⠠⠉

4.10 Affixing symbols to other symbols

`\overset`, `\underset`, and `\overunderset` are supported:

$\overset{*}{\Gamma}$ $\underset{\circ}{\Gamma}$ $\overset{*}{\underset{\circ}{\Gamma}}$ is `$\overset{*}{\Gamma}$` `$\underset{\circ}{\Gamma}$` `$\overunderset{*}{\circ}{\Gamma}$` and gives ⠨⠠⠛⠣⠈⠼⠻ ⠨⠠⠛⠩⠨⠡⠻ ⠨⠠⠛⠩⠨⠡⠣⠈⠼⠻

4.11 Fractions and related constructions

4.11.1 The `\frac`, `\dfrac`, and `\tfrac` commands

All these commands are supported:

$\frac{1}{n+2}$, $\dfrac{1}{n+2}$, and $\tfrac{1}{n+2}$ which is `$\frac{1}{n+2}$, $\dfrac{1}{n+2}$, and $\tfrac{1}{n+2}$` gives ⠹⠼⠂⠌⠝⠬⠼⠆⠼⠂ ⠹⠼⠂⠌⠝⠬⠼⠆⠼⠂ ⠁⠝⠙ ⠹⠼⠂⠌⠝⠬⠼⠆⠼

Notice that display and text fractions have the same output as there is no reason to differentiate them for the blind.

4.11.2 The `\binom`, `\dbinom`, and `\tbinom` commands

All these commands are supported:

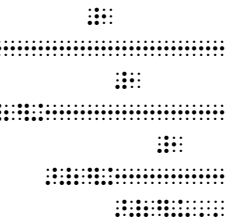
$2^k-\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}$ is `$2^k-\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}$` and gives ⠼⠆⠘⠅⠐⠤⠷⠅⠩⠼⠂⠾⠼⠆⠘⠅⠤⠼⠂⠐⠬⠷⠅⠩⠼⠆⠾⠼⠆⠘⠅⠤⠼⠆⠐

4.11.3 The `\genfrac` command

The `\genfrac` command relates only to visual issues so it is not supported.

4.12 Continued fractions

Continued fractions are supported using the `\cfrac` command. This is one of the few cases where latex2nemeth produces two-dimensional output:

Code	TEX output	Braille
`$$\cfrac{1}{\sqrt{2}+ \cfrac{1}{\sqrt{2}+ \cfrac{1}{\sqrt{2}+\dotsb }}}$$`	$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}}$	

4.13 Smash options

`\smash` is visual and ignored.

5 Delimiters

All sizing commands for delimiters are supported. Both `\left` and `\right` commands as well as all variants of `\big`.

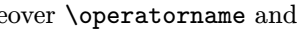
5.1 Vertical bar notations

All commands `\lvert`, `\rvert`, `\lVert`, `\rVert` are supported.

6 Operator names

6.1 Defining new operator names

`\DeclareMathOperator` and `\DeclareMathOperator*` are supported in the preamble. For example, placing `\DeclareMathOperator*{\Lim}{Lim}` in the preamble allows for

$\operatorname*{Lim}_n$ which is `$\Lim_n$` and gives 

Moreover `\operatorname` and `\operatorname*` in math formulæ are supported.

In addition to the above, predefined operator names are supported:

$\varprojlim$ which is `$\varprojlim$` and gives 

6.2 `\mod` and relatives

`\mod`, `\bmod`, `\pmod`, `\pod` also work:

$$\gcd(n, m \bmod n); \quad x \equiv y \pmod{b}; \quad x \equiv y \mod c; \quad x \equiv y \pod{d}$$

which is

```
$\gcd(n,m\bmod n);\quad x\equiv y\pmod b;
\quad x\equiv y\mod c;\quad x\equiv y\pod d$
```

gives

7 The `\text` command

The `\text` command is supported. For example:

$$\partial_s f(x) = \frac{\partial}{\partial x_0} f(x) \quad \text{for } x = x_0 + I x_1.$$

which is

```
$$\partial_s f(x) = \frac{\partial}{\partial x_0} f(x)\quad
\text{for $x= x_0 + I x_1$.}
$$
```

gives

8 Integrals and sums

8.1 Multiline subscripts and superscripts

Work has been done to support multiline subscripts and superscripts. Again, let's look at examples from the AMS documentation:

Code	TeX output	Braille

`$$\sum_{\substack{`
`0\le i\le m\\`
`0<j<n}}`
`P(i,j)`
`$$`
	$\sum_{\substack{0\le i\le m\\ 0<j<n}} P(i,j)$	⠨⠠⠎⠩⠼⠴ ⠐⠅ ⠚ ⠐⠅ ⠝⠩⠼⠴⠐⠅⠱⠊⠐⠅⠱⠍⠻⠠⠏⠷⠊⠠⠚⠾

Notice that the Braille substack is produced from bottom up. That is, $0 < j < n$ is written first and then $0 \le i \le m$, as is typical in the Nemeth standard.

Code	TeX output	Braille

`$$\sum_{\begin{subarray}{l}`
`i\in\Lambda\\ 0<j<n`
`\end{subarray}}`
`P(i,j)`
`$$`
	$\sum_{\begin{subarray}{l} i\in\Lambda\\ 0<j<n \end{subarray}} P(i,j)$	⠨⠠⠎⠩
		⠊ ⠈⠑ ⠨⠠⠇
		⠼⠴ ⠐⠅ ⠚ ⠐⠅ ⠝
		⠠⠏⠷⠊⠠⠚⠾

Here we notice that since an array is used the output is two-dimensional.

8.2 The `\sideset` command

The `\sideset` command is supported. An example:

Code	TeX output	Braille

`$$\sideset{}{'}`
`\sum_{n<k,\;\text{$n$ odd}}`
`nE_n`
`$$`
	$\sideset{}{'}\sum_{n<k,\;\text{n odd}} nE_n$	⠐⠨⠠⠎⠩⠝ ⠐⠅ ⠅⠠ ⠝ ⠕⠙⠙ ⠘⠄⠐⠻⠝⠠⠑⠰⠝⠐

Another example:

Code	TeX output	Braille
`$$\sideset{_*^*}{_*^*}\prod$$`	$\sideset{_*^*}{_*^*}\prod$	⠐⠰⠈⠼⠘⠈⠼⠐⠨⠠⠏⠰⠈⠼⠘⠈⠼⠐⠻

8.3 Placement of subscripts and limits

`\limits` and `\nolimits` are supported but `\displaylimits` is ignored as it is of no use to the blind.

8.4 Multiple integral signs

Multiple integral signs are all supported:

$\idotsint\limits_A$ which is `$$\idotsint\limits_A$$` gives ⠮⠄⠄⠄⠮⠩⠠⠁

9 Commutative diagrams

Commutative diagrams are not supported; they must be produced as tactile graphics.

10 Using math fonts

All `\mathbf`, `\mathsf`, `\mathcal`, `\mathrm`, `\mathsf`, `\mathtt` are supported.

11 A short guide for conversion to Braille and Nemeth

To convert TeX to Braille is impossible! There is a mathematical proof for this, but the short reason is the macro capabilities of TeX. So you can not convert arbitrary code to Braille. But

on the other hand you do not want to either, because many things are done for visual results that the blind do not need. So some minimal editing of the TEX file is unavoidable.

First, all pictures must be removed from the TEX file because pictures need another procedure to produce tactile graphics. However, latex2nemeth supports `pstricks`. So if your pictures are in the form

```
\begin{figure}[ht]
\begin{pspicture}(-2,-2)(2,3)
⟨ps picture commands⟩
\end{pspicture}
\caption{A picture}\label{mypic1}
\end{figure}
```

The program, while running, will:

- move the contents from `\begin{pspicture}` to `\end{pspicture}` to a separate file in your working directory,
- leave a comment in place of the figure to "see figure `label`",
- change all `pstricks` labels to Braille in the new picture files.

So while your file has been transcribed, you now have to modify the picture files it produced to give them proper characteristics for the blind. This part is discussed below with an example.

Assume now that we have a `file.tex` without any pictures in it. We start by simplifying the preamble. We should not have complicated macros. For example, running heads' configuration must be removed. It makes no sense for the blind. Customization of sections, chapter heads, etc., make no sense and must be removed.

Any `\tableofcontents` or similar is also removed; this needs some explanation. Braille files are not in a typeset format such as pdf files. They are simple text files. In order to predict the page of, say, the chapter of a book one needs to know how many lines will be embossed per page and how many braille characters per line. This information is not a standard. Embossers have different settings and it is only the driver of the embosser that could know this information. So a conversion program such as latex2nemeth cannot have access to such information. This is one of the reasons that the output of the program is split into chapters — to give the opportunity to the blind to organize in different folders (or to use tabs) the material of the book.

Latex2nemeth will not parse your `\usepackage` commands but will mostly ignore them. `\newtheorem` and simple `\newcommand` (with or without arguments) are supported. Finally the file must be in UTF-8 encoding. We now start the attempt to convert.

Run `xelatex` or `lualatex` in order to check that your file compiles and produces the `file.aux` file which is needed for the references mechanism.

Now run

```
latex2nemeth file.tex file.aux
```

Most of the time the first run will fail. Typically the user has forgotten to remove visual parts from the preamble. The program will inform you of the line and column of the problem it encountered. Fix it and re-run the above command. After enough corrections of your `.tex` file, the program will succeed. It will produce a `.nemeth` file for each chapter. These are plain text Braille files but in UTF-16 encoding. We need to convert them to UTF-8 and then either import them to LibreOffice for embossing or convert them to LibreOffice automatically.

Let's see the manual procedure first. Conversion to UTF-8 can be done with `iconv`:

```
iconv -f utf-16 -t utf-8 file0.nemeth > file0-u8.txt
```

Now convert to a LibreOffice `.odt` file:

```
libreoffice --headless --convert-to odt file0-u8.txt >/dev/null
```

This will produce `file0-u8.odt`. LibreOffice has a builtin default for the font. But we need a font that has Braille characters, such as DejaVu-Serif. So the final step is to open the `.odt` file, select the whole text (Control-a) and change the font to DejaVu Serif. Save the file.

LibreOffice has a plugin called `odt2braille`. This plugin must be installed in order to be able to drive the embosser. With the plugin installed, open the `odt` file and choose File→Emboss.

The whole process can be automated by a simple script such as this

```
#!/bin/sh
#get a random name first of 8 chars
tmpdir=`cat /dev/urandom | tr -cd 'a-f0-9' | head -c 8`

#make a folder
mkdir $tmpdir

#get the base name of the file to convert
file=`basename "$1" .nemeth`

#convert nemeth from utf16 to utf8
iconv -f utf-16 -t utf-8 "$1" >$file.txt

#convert txt file to odt
libreoffice --headless --convert-to odt $file.txt >/dev/null

# odt is setup with a bultin template for conversions
# from text that uses Liberation Mono font.
# we need DejaVu Serif. We change the font and repack
# the odt file.
unzip -qq -d $tmpdir $file.odt
rm -f $file.odt
find $tmpdir -type f | xargs sed -i 's/Liberation Mono/DejaVu Serif/g'
( cd $tmpdir; zip -qq -r ../$file.odt . )

#cleanup
/bin/rm -rf $tmpdir $file.txt
```

11.1 Conversion of pictures

Now let us turn to pictures. This is most of the work because we have to replace all labels in the picture with Braille (unless you used pstricks in which case the program automatically transcribes the labels) and make new placement decisions, since the Braille is usually long and will not fit in the original position of the label. The easy part is to make the picture lines wider so they can be detected by the hands of the blind. All lines should vary from 1.2 mm minimum to 1.8 mm. We can use this range to distinguish between logically different lines. For example, suppose we want to graph the function $f(x) = x^2$ from -2 to 2. The original graph may look like the one in Figure 1. The Braille for $f(x) = x^2$ is ⠍⠀⠭⠘⠼⠃ (we will come to this soon). We will change the axis width to 1.2 mm and the graph of the function to 1.8 mm. Since the file will be a pdf file produced in a tactile printer on micro-capsule paper, the Braille is not embossed. So we need to increase its character size to at least 24pt in order to be readable. Moreover the font must be a font such as `NewCMSans10-Book.otf` so that the Braille dots are for blind and not for sighted persons (as is the case with `NewCM10-Book.otf`). So the final graph will be as in Figure 2.

Finally we need an easy way to get the labels into Braille if we used a system other than `pstricks` for our graphics (e.g., `tikz`). An easy way, although time-consuming, is to use a command-line script for this. Create a script, say `l2n.sh`, with contents:

```
#!/bin/bash
echo "\documentclass{article}\usepackage{amsfonts}\begin{document}" \
     > ~/tmp/l2n.tex
echo "$1" >> ~/tmp/l2n.tex
```

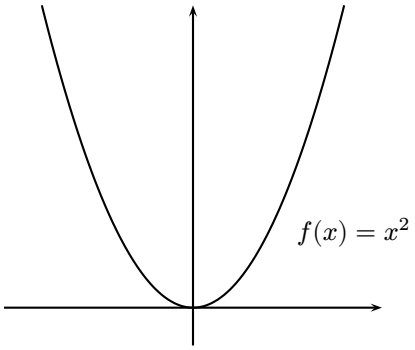


Figure 1: The $f(x) = x^2$ function for sighted persons.

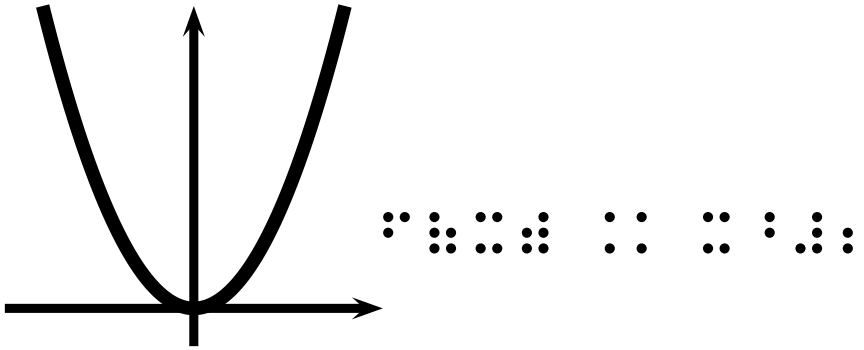


Figure 2: The $f(x) = x^2$ function for the blind

```
echo "\end{document}" >> ~/tmp/l2n.tex
touch ~/tmp/l2n.aux
cd ~/tmp/
latex2nemeth l2n.tex l2n.aux 2>/dev/null
iconv -f utf-16 -t utf8 l2n0.nemeth

echo " "
rm -f l2n.tex l2n0.nemeth l2n.aux
```

Then on the command-line, to get the Braille string for $f(x) = x^2$ run this

```
sh l2n.sh "\$f(x)=x^2\$"
```

and copy the output to your picture file at the proper label place. Produce the pdf file as you would normally (say with `xelatex`[1] or `lualatex`) and proceed to the tactile printer with micro-capsule paper. Pictures go as pdfs to tactile printers and the Braille text of the TEX files go as `odt` files to embossers.

12 Implementation

Latex2nemeth is written in Java using the JavaCC compiler construction tool. Its design is based on object-oriented techniques such as the Interpreter and Composite design patterns [1] for the representation of mathematical expressions. In order to support spatial aligned structures, as in the case of the `\cfrac` command, a two-dimensional buffer is created for every Braille expression, which is filled in a bottom-up fashion, so as to correctly calculate the dimensions of containing boxes, for example, the width and height of numerator and denominator in a fraction expression. In this way, a generic mechanism for two-dimensional structures was implemented. However, in expressions such as fractions (command `\frac`) which can be ex-

[1] You may need `xelatex-unsafe` if you are using `pstricks`

pressed in Nemeth code in both linear and two-dimensional arrangements, the current version of the program only provides the linear form of the output.

13 Symbols included in `unimath-symbols.pdf` but unsupported in Nemeth

`\arabicmaj`	⁦ـحـ⁩	arabic mathematical operator meem with hah with tatweel
`\arabichad`	⁦ـحـد⁩	arabic mathematical operator hah with dal
`\inttop`	⌠	top half integral
`\intbottom`	⌡	bottom half integral
`\varhexagonlrbonds`	⌬	six carbon ring, corner down, double bonds lower right etc
`\lparenuend`	⎛	left parenthesis upper hook
`\lparenextender`	⎜	left parenthesis extension
`\lparenlend`	⎝	left parenthesis lower hook
`\rparenuend`	⎞	right parenthesis upper hook
`\rparenextender`	⎟	right parenthesis extension
`\rparenlend`	⎠	right parenthesis lower hook
`\lbrackuend`	⎡	left square bracket upper corner
`\lbrackextender`	⎢	left square bracket extension
`\lbracklend`	⎣	left square bracket lower corner
`\rbrackuend`	⎤	right square bracket upper corner
`\rbrackextender`	⎥	right square bracket extension
`\rbracklend`	⎦	right square bracket lower corner
`\lbraceuend`	⎧	left curly bracket upper hook
`\lbracemid`	⎨	left curly bracket middle piece
`\lbracelend`	⎩	left curly bracket lower hook
`\vbraceextender`	⎪	curly bracket extension
`\rbraceuend`	⎫	right curly bracket upper hook
`\rbracemid`	⎬	right curly bracket middle piece
`\rbracelend`	⎭	right curly bracket lower hook
`\intextender`	⎮	integral extension
`\harrowextender`	⎯	horizontal line extension (used to extend arrows)
`\sumtop`	⎲	summation top
`\sumbottom`	⎳	summation bottom
`\sqrtbottom`	⎷	radical symbol bottom
`\lvboxline`	⎸	left vertical box line

`\rvboxline`		right vertical box line
`\elinters`		electrical intersection
`\blocklefthalf`		left half block
`\blockrighthalf`		right half block
`\circlelefthalfblack`		circle, filled left half [harvey ball]
`\circlerighthalfblack`		circle, filled right half
`\circlebottomhalfblack`		circle, filled bottom half
`\circletophalfblack`		circle, filled top half
`\circleurquadblack`		circle with upper right quadrant black
`\blackcircleulquadwhite`		circle with all but upper left quadrant black
`\blacklefthalfcircle`		left half black circle
`\blackrighthalfcircle`		right half black circle
`\invwhiteupperhalfcircle`		upper half inverse white circle
`\invwhitelowerhalfcircle`		lower half inverse white circle
`\ularc`		upper left quadrant circular arc
`\urarc`		upper right quadrant circular arc
`\lrarc`		lower right quadrant circular arc
`\llarc`		lower left quadrant circular arc
`\lrblacktriangle`		lower right triangle, filled
`\llblacktriangle`		lower left triangle, filled
`\ulblacktriangle`		upper left triangle, filled
`\urblacktriangle`		upper right triangle, filled
`\squareleftblack`		square, filled left half
`\squarerightblack`		square, filled right half
`\squareulblack`		square, filled top left corner
`\squarelrblack`		square, filled bottom right corner
`\triangleleftblack`		up-pointing triangle with left half black
`\trianglerightblack`		up-pointing triangle with right half black
`\squareulquad`		white square with upper left quadrant
`\squarellquad`		white square with lower left quadrant
`\squarelrquad`		white square with lower right quadrant
`\squareurquad`		white square with upper right quadrant
`\circleulquad`		white circle with upper left quadrant
`\circlellquad`		white circle with lower left quadrant
`\circlelrquad`		white circle with lower right quadrant
`\circleurquad`		white circle with upper right quadrant
`\ultriangle`		upper left triangle
`\urtriangle`		upper right triangle
`\lltriangle`		lower left triangle

`\lrtriangle`	◿	lower right triangle
`\quarternote`	♩	music note (sung text sign)
`\eighthnote`	♪	eighth note
`\twonotes`	♫	beamed eighth notes
`\iinfin`	∾	incomplete infinity
`\laplac`	□	square with contoured outline
`\downtriangleleftblack`	◭	down-pointing triangle with left half black
`\downtrianglerightblack`	◮	down-pointing triangle with right half black
`\squaretopblack`	⬒	square with top half black
`\squarebotblack`	⬓	square with bottom half black
`\squareurblack`	◹	square with upper right diagonal half black
`\squarellblack`	◺	square with lower left diagonal half black
`\diamondleftblack`	◖	diamond with left half black
`\diamondrightblack`	◗	diamond with right half black
`\diamondtopblack`	◆	diamond with top half black
`\diamondbotblack`	◆	diamond with bottom half black
`\mttzero`	0	mathematical monospace digit 0
`\mttone`	1	mathematical monospace digit 1
`\mtttwo`	2	mathematical monospace digit 2
`\mttthree`	3	mathematical monospace digit 3
`\mttfour`	4	mathematical monospace digit 4
`\mttfive`	5	mathematical monospace digit 5
`\mttsix`	6	mathematical monospace digit 6
`\mttseven`	7	mathematical monospace digit 7
`\mtteight`	8	mathematical monospace digit 8
`\mttnine`	9	mathematical monospace digit 9

References

[1] E. Gamma, R. Helm, et al. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley, Boston, MA, USA, 1994.

◇ Andreas Papasalouros
University of the Aegean
Department of Mathematics
832 00 Karlovassi
Samos, Greece
http://www.samos.aegean.gr/math/andpapas/cv_en.html

◇ Antonis Tsolomitis
University of the Aegean
Department of Mathematics
832 00 Karlovassi
Samos, Greece
http://myria.math.aegean.gr/~atsol