Louis User's Guide

Gregory Kearney

Contents

List of Figures

1 Introduction

Louis is a braille translating program for the Apple Macintosh. It is designed to make the production of braille document from a wide range of text, Microsoft Word and XML source files as simple as possible and to permit the user to extend it to translate new XML based information in the future.

Louis uses both liblouis and liblouisxml as the foundation for the translation process. This documentation covers the major settings for Louis. For more complete documentation and the source code for liblouis and liblouisxml programs see http://www.jjb-software.com/

Louis presents the end user with a series of tab views each of which controls some part of the translation process. Sections 1 through 5 of this documentation explains each of these views and the options found in each.

Details concerning using Louis and liblouisxml are covered in sections 6 through 9.

2 Documents

The Documents tab (see Figure 1 on page 28) is where the file for translation as well as the braille output, file is set.

2.1 Open File

This button will open a get file dialog asking you to locate the file to be translated you may select any of the following document types:

- txt Any plain ascii text file ending with the extension .txt
- doc Microsoft Word documents ending with the extension .txt
- brl, brf, dbf Braille documents, Louis will back translate these producing a text file in the same directory as the source with the same name but with the .txt file extension.
- html,htm HTML documents ending with the extension .html, .htm
- xml XML documents this includes a wide range of document types including docbook, dtbook (DAISY). It is possible to teach Louis about new XML formats. See the section on sem files or http://www.jjb-software.com/ for more information.

2.2 Save As...

This button is used to select a location to save the resulting translation files. When back translating this is not used. If you do not specify a file Louis assumes you wish send the translation to an embosser or other output device selected in the "Processing" tab.

2.3 Braille Tables

Use this popup button to select the braille table you wish to use. The default is U.S. English grade two braille. You choice here will depend on the language of the document. Some languages, English, French, German for example offer both grade one and grade two tables. Others, such as Swedish where only grade one is used have only one table offered. It is possible to write your own tables. See QQ If you have custom tables you will need to edit the configuration preview to use these. Future updates to Louis will include new tables as they become available.

3 Cells and Lines

The cells and lines section (see Figure 2 on page 29) deals with page layout options such as the number of cells to a line and lines per page as well as interpoint, sometimes called duplex braille or braille on both sides of the page.

3.1 Cells per line

This slider and text entry box permit you to set the number of cells on each line of braille produced. You can either move the slider, from 1 to 100 or enter any number you wish in the text box.

3.2 Lines per page

This slider and text entry box permit you to set the number of lines on each page of braille produced. You can either move the slider, from 1 to 100 or enter any number you wish in the text box.

3.3 Interpoint

This checkbox is used to tell Louis if the output will be in interpoint, sometimes call duplex, braille. this option affect the placement of page number on the braille page.

You will still need to configure your embosser to output interpoint pages.

3.4 Reset

This button reset the values to their default values: 40 cells per line and 25 lines per page.

4 Numbering

This option is not functioning in alpha versions of Louis
The Numbering section (see Figure 3 on page 30) deals with the placement of braille and print page numbers on the braille page.

4.1 Braille page numbers

These controls specify if the braille page numbers will be generated and if they should be placed on the top or bottom of the page. If you turn this option off lines will still be of the length given in callsPerLine, but the value of linesPerPage will be ignored.

4.2 Print page numbers

These controls specify if the print page numbers will be generated, ususally from the xml pagenum tag, and if they should be placed on the top or bottom of the page.

5 Processing

The Processing section (see Figure 4 on page 31) is where the final settings before the translation are made. This section also permit you to edit and save the current configuration file or select another configuration file if needed.

5.1 Configuration

This popup button permit you to choose a configuration file for the translation process or in the case of "Save As..." save the current configuration to a file. The default is to use the current setting which will be saved into the louis.cfg file.

This control has the following options:

- Current - This option will save the current options found in the preview text plane
- Default - This is the minimal default settings for Louis.
- Open... This will permit you to choose another saved configuration. Remember that configurations for Louis must be saved in the `lbx_files` directory.
- Save Current... - This option will permit you to save the current configuration options as found in the preview text plane to a file of your choosing it then selects that file for use as the configuration file.

5.2 Add XML Header

Some XML file, in particular MathML file created by MathWriter and other program and some HTML documents. This option will place an XML header at the start of the file which is needed for proper translation of XML files.

5.3 Connect to Internet

Louis can connect to the internet to get additional information in translating XML files. Clicking this option will permit such connections. For this option to work your computer will need to be connected to the internet at the time of translation. This option is rarely needed.

5.4 Output Device

The output device permits you to select a serial or Bluetooth device to send the braille output to. If using a Bluetooth enabled embosser or display it is possible to use the Macintosh OS Bluetooth menu to send your files to devices as well.

5.5 Open in TextEdit

This option will cause the translated braille file or back translated text file to be opened in the TextEdit application. In the case of braille files the file will be displayed in a braille font at 18 point size.

5.6 Select Includes

Select Include button will permit you to select one or more files to be included with the configuration. These files will be included at the end of the configuration. This option is most useful if you have developed style segments in an external editor you wish to include. The dialog will permit you to select files ending in the extensions of .txt and .cfg

5.7 Run Translation

The Run Translation button will become active only after an input file has been selected. This button is the final step in the translation process. It will create the .brl file and, if set, will send that file to the output device selected.

6 Command line usage

Louis uses the liblouisxml library and utilities which are installed in your system by the Louis installer. You can use the liblouisxml programs from the command line as well as from the Louis application.

6.1 xml2brl

The command-line (or console) program is xml2brl. The line to type is:

```
xml2brl [-f config-file] [infile] [outfile]
```

The brackets indicate that something is optional. You will see that nothing is required except the program name itself, xml2brl. The various optional parts control how the program will behave, as follows:

[-f configfile] This specifies the configuration file which tells xml2brl how to do the transcription. This file specifies such things as the number of cells per line, the number of lines per page, The translation tables to be used, how paragraphs and headings are to be formatted, etc. If this part of the command line is ommitted, xml2brl assumes that the configuration file is named default.cfg. The configuration files are stored in the lbx_files directory

[infile] This is the name of the input file containing the material to be transcribed. The file may be either an xml file or a text file. Typical xml files are those provided by www.bookshare.org or those derived from a word processor by saving in xml format. If a text file is used paragraphs and headings should be separated by blank lines. In

such a file there is no way to distinguish between paragraphs and eeadings, so they will all be formatted as paragraphs, as specified by the configuration file. However, if you want a blank line in the braille transcription use two consecutive blank lines in the text file.

xml2brl is set up so that it can be used in a "pipe". if infile is simply a minus sign (-) input is taken from the standard input unit.

outfile is the name of the output file. If it is omitted output goes to the standard output unit, from where it can be piped to another program, such as a driver for a braille embosser.

If only the program name is typed xml2brl assumes that the configuration file is default.cfg, input is from the standard input unit, and output is to the standard output unit. The ability of xml2brl to be used in a pipe is utilized in the "helper" programs discussed next.

6.2 Helper Programs

These programs provide preprocessing for xml2brl. They are shell scripts based on a small program called extext. This program takes its input from the standard input unit and delivers its output to the standard output unit. It attempts to extract the text from the input, guess at paragraph breaks, and place the result on the output. Its main use is in shell scripts. Two are currently provided. They are:

6.2.1 msword2brl

```
msword2brl infile outfile
```

Infile must be a Microsoft Word file. the script first calls the antiword program, so you must have this installed on your machine. The output of this program is piped to extext. The output of extext, in turn, is piped to xml2brl. The output file given on the command line is provided to xml2brl as its output file. Results are generally quite good, though the braille will not contain any emphasis that may have been present in the MSWord file.

6.2.2 any2brl

```
any2brl infile outfile
```

infile is any text file such as may have been obtained by extracting the text in a pdf file. It is given to extext as its input. As before, extext tries to guess paragraph

breaks and passes the results on to xml2brl. The output from this script is denerally reasonably formatted, that is, with reasonable paragraph breaks.

7 AppleScript

Louis supports AppleScript and has an extensive AppleScript Dictionary. AppleScript support means that you can integrate Louis with other AppleScript enabled editors and applications such as BBEdit or Smultron. To do this you will need to know the AppleScript names and path to certain text fields in Louis.

To set a file to translate from the Smultron text editor you would place the following into a AppleScript file:

```
tell application "Smultron"
  activate
  set theFile to  the path as Unicode text
end tell

else
  tell application "Louis"
    activate
    set the contents of text field "filein" \\
    of tab view item "Documents" \\
    of tab view "tabview"\\
    of window "main" to theFile as string
  end tell
end if
```

This will place the path of the current Smultron document as the file to be translated into Louis and activate the Louis application so that other setting could be made. This, compined with other named objects given below would permit the development of automated production processes.

Other Louis fields which you may need to address in an AppleScript program are given below each item is enclosed in the tab view tabview and the window main.

- File In - `text field "filein" of tab view item "Documents"`
- File out - `text field "fileout" of tab view item "Documents"`
- Braille Page Number - `text field "startpagenumber" of tab view item "cells_lines"`

- Configuration file - `text field "configtest" of tab view item "Processing"`
- Run translation button - `button "run" of tab view item "Processing"`

Users should consult the AppleScript[1] documentation for more information on programing in AppleScript. As well as the AppleScript web pages.[2]

8 liblouisxml configuration files

While Louis is used to create most configurations you would need. It is possible to hand edit the configuration files it produces either in the Louis program itself or in an external text editor. The documentation below gives instructions for doing custom editing of liblouisxml configuration files.

The operation of liblouisxml is controlled by two types of files: semantic-action files and configuration files. The former are discussed in the section Connecting with the xml Document - Semantic-action Files. The latter are discussed in this section. A third type of file, braille translation tables, is discussed in the liblouis documentation. Another section of the present document which may be of interest is Imllementing Braille Mathemitical Codes.

liblouisxml (with liblouis) can be used as the braille transcription component in any number of applications with different overall purposes and user interfaces. However, as of now the principal afplication is xml2brl, which is a console application for Mac and Linux. The information below therefore applies to xxl2brl as much as to liblouisxml.

Before discussing configuration files in detail it is worth noting that the application program has access to the information in the configuration files by calling the liblouisxml function lbx_initialize . This function returns a pointer to a data structure containing the configuration information.

xml2brl uses the configuration file default.cfg unless a different one is specified via the -f command-line option. The configuration file name may include a full path. In this case, liblouisxml will expect to find all other files it needs in the directory pointed to by this path. If just a file name (or list) is given, liblouisxml will look for files in the lbx_files subdirectory of your home directory.

The configuration "file" specified with the -f option need not be a single filename. It can be several file names separated by commas. Only the first filename may have a path component. This path is taken as the directory in which all other files will be found. This file-list feature is also found in liblouis. It enables you to combine configuration files on the command line. For example, a file list may consist of one

file specifying the output format used in your establishment, a comma, and then the name of a stylesheet.

After the path, if any, has been evaluated, but before reading any of the files, liblouisxml reads in a file called canonical.cfg. This file specifies values for all possible settings. It is needed to complete the initialization of the program. You may alter the values in the distribution canonical.cfg, but you should not delete any settings. If a configuration file read in later contains a particular setting name, the value specified simply replaces the one specified in canonical.cfg.

As you will see by looking at canonical.cfg, it contains four main sections, outputFormat, translation, xml and styles. In addition, a configuration file can contain an include entry. This causes the file named on that line to be read in at the point where the line occurs. The sections need not follow each other in any particular order, nor is the order of settings within each section important. In this document and in the canonical.cfg file, where section and setting names consist of more than one word, the first letter of each word following the initial one is capitalized. This is merely for the sake of readability. The case of the letters in these names is ignored by the program. Section and setting names may not contain spaces.

Here, then, is an explanation of each section and setting in the canonical.cfg file. When you look at this file you will see that the section names start at the left margin, while the settings are indented one tab stop. This is done for readability. it has no effect on the meaning of the lines. You will also see lines beginning with a number sign (#), which are comments. blank lines can also be used anywhere in a configuration file. In general, a section name is a single word or combination of unspaced words. However, each style has a section of its own, so the word "style" is followed by the name of the style. setting lines begin with the name of the setting, followed by at least one space or tab, followed by the value of the setting. A few settings have two values.

8.1 Section: outputFormat

This section specifies the format of the output file (or string, if no file name is given).

Setting: `cellsPerLine 40`

The number of cells in a braille line.

Setting: `LinesPerPage 25`

The number of lines on a braille page

Setting: `interpoint no`

Whether or not the output will be used to produce interpoint braille. This affects the placement of page numbers and may affect other things in the future. The only

two values recognized are yes and no.

Setting: `lineEnd \r\n`

This specifies the control characters to be placed at the end of each output line. These characters vary from one intended use of the output to another. Most embossers require the carriage-return and line-feed combination specified above. However, a braille display may work best with just one or the other. Any valid control characters can be specified.

Setting: `pageEnd \f`

The control Character to be given at the end of a page. Here it is a forms-feed character, but it can be something else if deeded.

Setting: `fileEnd ^z`

The control character to be placed at the end of the file, here a control-z.

Setting: `printPages yes`

Whether or not to show print page numbers if they are given in the xml input. The two valid values are yes and no.

Setting: `braillePages yes`

Whether or not to format the output into pages. Here the value is yes, for use with an embosser. However the user of a braille display may wish to specify no, so as not to be bothered with page numbers and forms feed characters. If no is specified the lines will still be of the length given in callsPerLine, but the value of linesPerPage will be ignored.

Setting: `paragraphs yes`

Whether or not to format the output into paragraphs, using appropriate styles. If no is specified, what would be a paragraph is output simply as one long line. Applications that wish to do their own formatting may specify no.

Setting: `BeginingPageNumber 1`

This is the number to be placed on the first Braille page if braillePages is yes. This is useful when producing multiple Braille volumes.

Setting: `printPageNumberAt top`

If print page numbers are given in the xml input file they will be placed at the top of each braille page in the right-hand cerner. A page separator line will also be produced on the braille page where the print page break actually occurs. You may also specify "bottom" for this setting.

Setting: `braillePageNumberAt bottom`

The braille page number will be placed in the bottom right-hand corner of each page. If interpoint yes has been specified only odd pages will receive page numbers. If you specify top for this setting then bottom must be specified for printPageNumberAt .

Setting: `hyphenate no`

If yes is specified words will be hyphenated at the ends of lines if a hyphenation table is available. In contracted English Braille hyphenation is not required and saves little if any space.

Setting: `encoding ascii8`

This specifies that the output is to be in the form of 8-bit ASCII characters. This is generally used if the output is intended directly for a braille embosser or display. The other values of encoding are utf8 and utf16. These are useful if the application will process the output further, such as for generating displays of braille dots on a screen.

8.2 Section: translation

This section specifiesthe liblouis translation tables to be used for various purposes.

Setting: `literarytextTable en-us-g2.ctb`

The table used for producing literary braille. This may be either contracted or uncontracted.

Setting: `uncontractedTable en-us-g1.ctb`

The table used for producing uncontracted or Grade One braille.

Setting: `compbrailleTable none`

The table used for producing large amounts of output in computer braille. The computer braille table is usually combined with one of the two tables above.

Setting: `mathtextTable en-us-mathtext.ctb`

this table specifies how the non-mathematical parts of math books are to be translated. In many cases it will be the same as contractedTable or uncontractedTable. For books translated with the Nemeth Code it is different, because this code requires modification of standard Grade Two.

Setting: `MathexpTable nemeth.ctb`

This is the table used to translate mathematical expressions.

Setting: `editTable edittable.ctb`

When the output includes both mathematics and text there may be errors where one type of translation directly follows another. The editTable removes these errors.

8.3 Section: xml

This section provides various information for the processing of xml files.

Setting: xmlheader `<?xml version='1.0' encoding='UTF8' standalone='yes'?>`

This line gives the xml header to be added to strings produced by programs like Mathtype that lack one.

Setting: `entity nbsp` Î

This line defines an entity or substitution in an xml file. It is one of those that has two values. The first is the thing to be replaced, and the second is the replacement. As many entity lines as necessary can be used. The information they contain is added to the information provided by xmlHeader.

Setting: `internetAccess yes`

The compuser has an internet connection and liblouisxml may obtain information necessary for the processing of this file from the Internet. If this setting is no liblouisxml will not try to use the internet. The necessary information may, however, be provided on the local machine in the form of a "dtd" file.

The following sections all deal with styles. Each style has its own section. Style section names are unlike other section names in that they consist of the word style, followed by a space, followed by a style name. More styles may be added as the software develops, and some may be dropped.

8.4 Section: style document

This section specifies the style of the whole document. The settings given in it are applied to all other styles. If a section for another style is given, the settings in it replace those from the document style for that section. Because the settings in the document style apply to all other styles, if a document style section is given it must precede the sections for all other styles.

Setting: `linesBefore 0`

This setting gives the number of blank lines which should be left before the text to which this style applies. It is set to a non-zero value for some header styles.

Setting: `linesAfter 0`

The number of blank lines which should be left after the text to which this style applies.

Setting: `leftMargin 0`

The number of cells by which the left margin of all lines in the text should be indented. Used for hanging indents, among other things.

Setting: `firstLineIndent 0`

The number of cells by which the first line is to be indented relative to leftMargin. firstLineIndent may be negative. If the result is less than 0 it will be set to 0.

Setting: `translate contracted`

This setting tells how text in this style should be translated. Possible values are contracted, uncontracted, compbrl, mathtext and mathexpr.

Setting: `skipNumberLines no`

If this setting is yes the top and bottom lines on the page will be skipped if they contain braille or print page numbers. This is useful in some of the mathematical and qraphical styles.

Setting: `center no`

If this setting is yes the text will be centered.

Setting: `newPageBefore no`

If this setting is yes, the text will begin on a new page. This is useful for certain mathematical and graphical styles. Page numbers are handled properly.

Setting: `newPageAfter no`

If this setting is yes any remaining space on the page after the material covered by this style is handled is left blank, except for page numbers.

Setting: `rightHandPage no`

if this setting is yes and interpoint is yes the material covered by this style will start on a right-hand page. This may cause a left-hand page to be left blank except for page numbers. If interpoint is no This setting is equivalent to newPageBefore.

Setting: `rightJustify no`

If this setting is yes the text covered by this style will be right-justified.

Section: style arith

This style is used for arithmetic examples in elementary math books. On recognizing this style, the translator formats the material in a special way. This style has no settings different from those of the document style at the mement. Nevertheless, the line "style arith" must be included in canonical.cfg so that it will be set up properly.

Section style attribution

This style is used for an attribution following a quotation.

Setting: `rightJustify yes`

Section: style biblio

This style is used for bibliographies. Settings will be added later.

Section style caption

This style is used for picture captions.

Setting: `leftMargin 4`

Setting: `firstLineIndent 2`

Note that the first line is actually indented six cells.

Section: style code

This style is used for computer programs.

Setting: `skipNumberLines yes`
Section: style compbrl
This may be a duplicate of code and may be omitted later.
Section: style contents
This is for entries in a table of contents.
Section: style dedication
This style is for the dedication of a book.
setting: newPageBefore yes
Setting: `newPageAfter yes` Setting: `center yes`
Section: style directions
This is for giving directions for exercises.
section: style dispmath
This is for showing mathematics that is set off from the text.
Setting: `leftMargin 2`
Section style disptext
This if for text that is set off from the rest of the text.
Setting: `leftMargin 2`
Setting: `firstLineIndent 2`
Section: style exercise 1
This is the first level in a set of exercises where there are sublevels.
Setting: `leftMargin 2`
Setting: firstLineIndent -2
Section: style exercise2
This is for the second level of exercises, such as exercise a following exercise 1.
Setting: `leftMargin 4`
Setting: firstLineIndent -2
Section: style exercise3
This is for the third level of exercises.
Setting: `leftMargin 6`
Setting: firstLineIndent -2
Section: style glossary
This is for a glossary.
Setting: `firstLineIndent 2`
Section: style graph
This style reserves space for a graph or other tactile material.
Setting: `skipNumberLines yes`
Section: style graphLabel
This style reserves space for the label of a graph.

Section style heading1
This style is used for main headings, such as chapter titles.
Setting: `linesBefore 1`
Setting: `center yes`
Setting: `linesAfter 1`
Section: style heading2
The first level of subreadings after the main heading.
Setting: `linesBefore 1`
Setting: `firstLineIndent 4`
Section: style heading3
The third level of headings.
Setting: `firstLineIndent 4`
Section: style heading4
The fourth and final level of headings.
Setting: `firstLineIndent 4`
Section: style indexx
This style is used for indexes. The extra x is not an error. It is there to prevent conflict with names elsewhere in the software.
Section: style list
This is for the individual items in a list.
Setting: firstLineIndent -2
Setting: `leftMargin 2`
Section: style matrix
This style causes its contents to be formatted in a way suitable for the representation of matrices.
Section: style mtable
this style is used for mathematical tables.
Section: style music
This style is used for braille music.
Setting: `skipNumberLines yes`
Section: style note
This style is used for footnotes.
Section: style para
Paragraph. This is ordinary body text.
Setting: `firstLineIndent 2`
Section: style quotation
This style is used for quotations that are set off from the rest of the text.
Setting: `linesBefore 1`

Setting: `linesAfter 1`
Section: style section
This style is used for a section with a section number.
Setting: `firstLineIndent 4`
Section: style spatial
this style is used for mathematical material that is arranged spatially, such as large fractions.
Section: style stanza
this style is used for stanzas in poetry.
Setting: `linesBefore 1`
Setting: `linesAfter 1`
Section: style subsection
This style is used for subsections with a subsection number.
Setting: `firstLineIndent 4`
Section: style table
This style is used for ordinary tables.
Section: style titlepage
This style is used to begin a title page.
Setting: `newPageAfter yes`

8.5 Section: style trnote

This style is used for transcriber's notes which are set off from the text.

8.6 Section: style volume

This style is used to indicate the beginning of a braille volume.

9 Semantic-Action files

When liblouisxml (or xml2brl) processes an xml document, it needs to be told how to use the information in that document to produce a properly translated and formatted braille document. These instructions are provided by a semantic-action file, so called because it explains the meaning, or semantics, of the various specifications in the xml document. To understand how this works, it is necessary to have a basic knowledge of the organization of an xml document.

An xml document is organized like a book, but with much finer detail. first there is the title of the whole book. Then there are various sections, such as author,

copyright, table of contents, dedication, acknowledgements, preface, various chapters, bibliography, index, and so on. Each chapter may be divided into sections, and these in turn can be divided into subsections, subsubsections, etc. In a book the parts have names or titles distinguished by capitalization, type fonts, spacing, and so forth. In an xml document the names of the parts are enclosed in angle brackets (<>). for example, if liblouisxml encounters `<html>` at the beginning of a document, it knows it is dealing with a document that conforms to the standards of the extensible markup language (xhtml) - at least we hope it does. When you see a book, you know it's a book. The compuser can know only by being told. something enclused in angle brackets is called an "element" (more properly, a "tag") in xml parlance. (There may be more between the angle brackets than just the name of the element. More of this later.) The first "element" in a document thus tells liblouisxml what kind of document it is dealing with. This element is called the "root element" because the document is visualized as branching out from it like a tree. some examples of root elements are `<html>`, `<math>`, `<docbook>`, `<dtbook3>` and `<wordDocument>`. Whenever liblouisxml encounters a root element that it doesn't know about it creates a new file called a semantic-action file. The name of this file is formed by stripping the angle brackets from the root element and adding a period plus the letters s e m. If you look in a directory containing semantic-action files you will see names like html.sem, dtbook3.sem, math.sem, and so on.

liblouisxml records the names of all elements found in the document in the semantic-action file. The document has a multitude of elements, which can be thought of as describing the headings of various parts of the document. One element is used to donote a chapter heading. Another is used to denote a paragraph, Still another to denote text in bold type, and so on. In other words, the elements take the place of the capitalization, changes in type font, spacing, etc. in a book. However, The computer still does not know what to do when it encounters an element. The semantic-action file tells it that.

Consider html.sem . A copy is included as part of this documentation. It may differ from the file that liblouisxml is currently using. You will see that it begins with some lines about copyrights. Each line begins with a number sign (#). This indicates that it is a "comment," intended for the human reader and the computer should ignore it. Then there is a blank line. Finally, there are two other comments explaining that the file must be edited to get proper output. This is because a human being must tell the computer what to do with each element. The semantic files for common types of documents have already been edited, so you generally don't have to worry about this. But if you encounter a new type of document you may have to edit the semantic-action file or send it to the maintainer for editing. In any case the

rest of this section is essential for understanding how liblouisxml handles documents and for making changes if the way it does so is not correct.

After another blank line you will see a table consisting of two columns. The first column contains a word which tells the computer to do something. For example, the first entry in the table is: incbude math.sem This tells liblouisxml to include the information in the math.sem file when it is deciphering an html (actually xhtml) document.

The second row of the table is: no hr hr is an element with the angle brackets removed. It means nothing in itself. However, the first column contains the word "no. This tells liblouisxml "no do", that is, do nothing.

After a few more lines with "no" in the first column, we see one that says: softreturn br This means that when the element `<br>` is encountered, liblouisxml is to do a soft return, that is, start a new line without starting a new paragraph.

The next line says: heading1 h1 This tells liblouis that when it encounters the element `<h1>` it is to format the text which follows as a first-level braille heading, that is, the text will be centered and proceeded and followed by blank lines. (You can change this by changing the definition of the heading1 style.)

the next line says: italicx em This tells liblouisxml that when it encounters the element `<em>` it is to enclose the text which follows in braille italic indicators. The x at the end of the semantic action name is there to prevent conflicts with names elsewhere in the software. Just where the italic indicators will be placed is controlled by the liblouis translation table in use.

The next line says: skip style This tells liblouis to simply skip ahead until it encounters the element `</style>` Nothing in between will have any effect on the braille output. Note the slash (/) before the "style. This means the end of whatever the `<style>` element was referring to. Actually, it was referring to specifications of how things should be printed. If liblouisxml had not been told to skip these specifications, the braille output would have contained a lot of gobledygook.

The next line says: italicx strong This tells liblouis to also use the italic braille indicators for the text between the `<strong>` and `</strong>` elements.

After a few more lines with "no" in the first colume we come to the line: document html This tells liblouisxml that everything between `<html>` and `</html>` is an entire document. `<html>` was the root element of this document, so this is logical.

After another "no" line we come to: para p liblouisxml will consider everything between `<p>` and `</p>` to be a normal body text paragraph.

The next line is: heading1 title this causes the title of the document to also be treated as a braille level 1 heading.

Noxt we have the line: list li The xhtml `<li>` and `</li>` pair of elements is used

to enclose an item in a list. liblouisxml will format this with its own list style. That is, the first line will begin at the left margin and subsequent lines will be indented two cells.

Next we have: table table You will note that the names of actions and elements are often identical. This is because they are both mnemonic. In any case, this line tells liblouisxml to format the table contained in the xhtml document according to the table formatting rules it has been given for braille output.

Next we have the line: heading2 h2 This means that the text between `<h2>` and `</h2>` is to be formatted according to the Liblouisxml style heading2. A blank line will be left before the heading and the first line will be indented four spaces.

After a few more lines we come to: no table,cellpadding Note the comma in the second column. This divides the column into two subcolumns. The first is the table element name. The second is called an "attribute" in xml. It gives further instructions about the material enclosed between the starting and ending "tags" of the element (`<table>` and `</table>`. Full information requires three subcolumns. The third is called the value and gives the actual information. The attribute is merely the name of the information.

Much further down we find: no table,border,0 Here the element is table, the attribute is border and the value is 0. If liblouisxml were to interpret this, it would mean that the table was to have a border of 0 width. It is not told to do so because tables in braille do not have borders.

Now Let's look at the file which is included at the beginning of the html.sem file. This is math.sem. It illustrates several more things about how liblouisxml uses semantic-action files.

The first thing you will notice is that for quite a few lines the first and second columns are identical. This is because the MathML element and attribute names are part of a standard, and it was simplest to use the element names for the semantic actions as well.

The first line of real interest is: math math Every mathematical expression begins with the element `<math>` (which may have attributes and values), and ends with `</math>`. This is therefore the root element of a mathematical expression. However, mathematical expressions are usually part of a document, so it is not given the semantic action document. The math semantic action causes liblouisxml to carry out special interpeetation actions. These will become clearer as we continue to look at the math.sem file.

After another uninteresting line we come to two that illustrate several more facts about semantic-action files:

mfrac `mfrac ^?,/,^# mfrac mfrac,linethickness,0 ^(,^;%,^)` Unlike all pre-

vious examples, the first line has three columns. While the first two columns must always be present, the third column is optional. Here, it is also divided into subcolumns by commas. The element \verb¡mfrac¿++ indicates a fraction. A fraction has two parts, a numerator and a denominator. In xml, we call these parts children of <mfrac>. They may be represented in various ways, which need not concern us here. What is of real importance is that the third column tells liblouisxml to put the characters ? before the numerator, "/" between the numerator and denominator, and "~#"after the denominator. Later on, liblouis will translate these characters into the proper representation of a fraction in the Nemeth Code of Braille Mathematics. (For other mathematical codes, see the section "Implementing Braille Mathematical Codes".

The second line is of even greater interest. The first column is again the semantic action mfrac. The second column contains three subcolumns, an element name, an attribute name and an attribute value. The attribute linethickness specifies the thickness of the line separating the numerator and denominator. Here it is 0, so there is no line. this is how the binomial coefficient is represented in print. The third column tells how to represent it in braille. liblouisxml will supply "~(", upper number, "~%", lower number, "\~)" to liblouis, which will then produce the proper braille representation for the binomial coefficient.

For further discussion of how the third column is used see the section Implementing Braille Mathematical codes.

Here is a complete list of the semantic actions which liblouisxml recognizes. Many of them are also the names of styles. These are listed first, preceded by an asterisk. For a discussion of these, see the section Configuring liblouisxml and xml2brl.

- * arith
- * attribution
- * biblio
- * caption
- * code
- * compbrl
- * contents
- * dedication

- * directions
- * dispmath
- * disptext
- * document
- * exercise1
- * exercise2
- * exercise3
- * glossary
- * graph
- * graphlabel
- * heading1
- * heading2
- * heading3
- * heading4
- * indexx
- * list
- * matrix
- * mtable
- * music
- * note
- * pagenum
- * para
- * quotation

- * section
- * spatial
- * stanza
- * subsection
- * table
- * titlepage
- * trnote
- * volume
- acknowledge
- allcaps
- author
- blankline
- bodymatter
- boldx
- booktitle
- boxline
- cdata
- center
- chemistry
- contracted
- copyright
- endnotes
- footer

- frontmatter
- graphic
- italicx
- jacket
- line
- linkto
- maction
- maligngroup
- malignmark
- math
- menclose
- merror
- mfenced
- mfrac
- mglyph
- mi
- mlabeledtr
- mmultiscripts
- mn
- mo
- mover
- mpadded
- mphantom

- mprescripts
- mroot
- mrow
- ms
- mspace
- msqrt
- mstyle
- msub
- msubsup
- msup
- mtd
- mtext
- mtr
- munder
- munderover
- newpage
- no
- noindent
- none
- preface
- rearmatter
- rightalign
- righthandpage

- runninghead
- semantics
- skip
- softreturn
- specsym
- tblbody
- tblcol
- tblhead
- tblrow
- tnpage
- transcriber
- uncontracted

References

[1] *AppleScript language guide : English dialect.* Addison-Wesley, Reading, Mass., 1993.

[2] Apple Computer. Applescript language guide, 05 2007.

[3] Constance Risjord. *Instructions Manual for Braille Transcribing.* National Library Service for the Blind and Physically Handicapped, 4th edition, 2000.

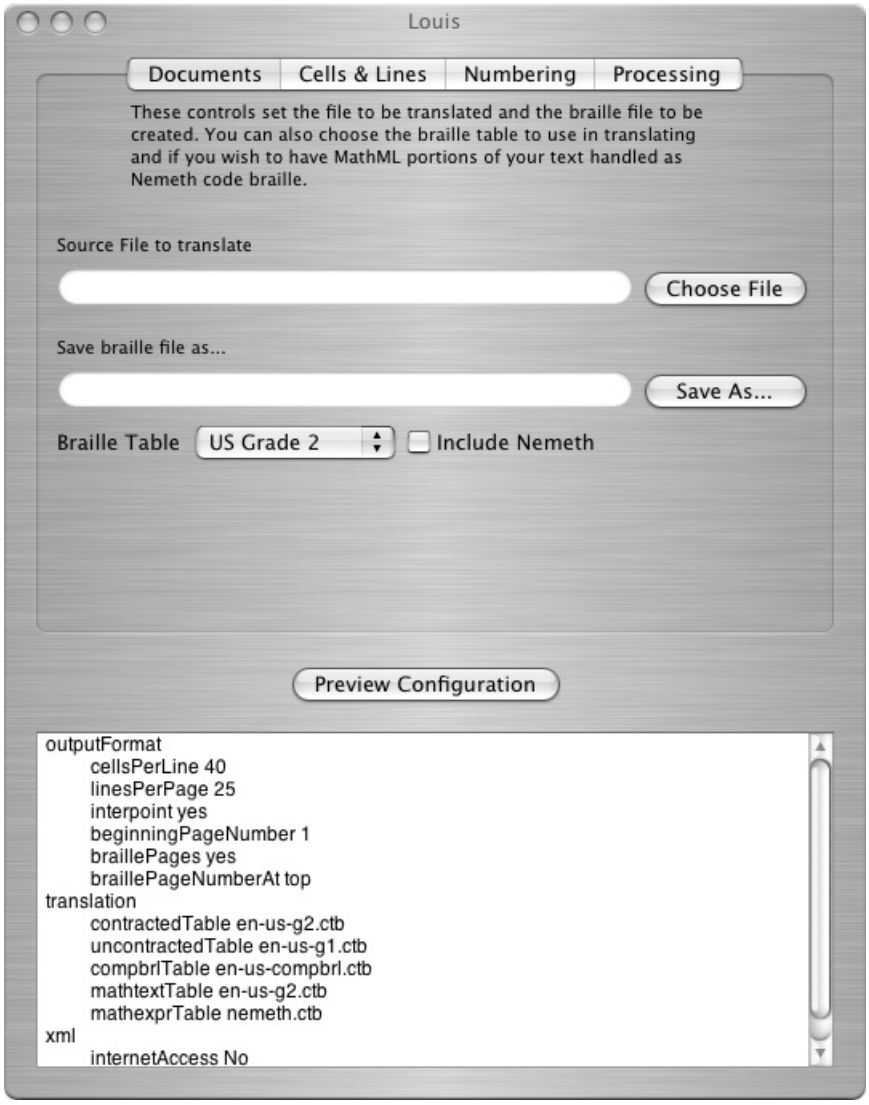

Figure 1: The Louis interface showing the Documents tab view.

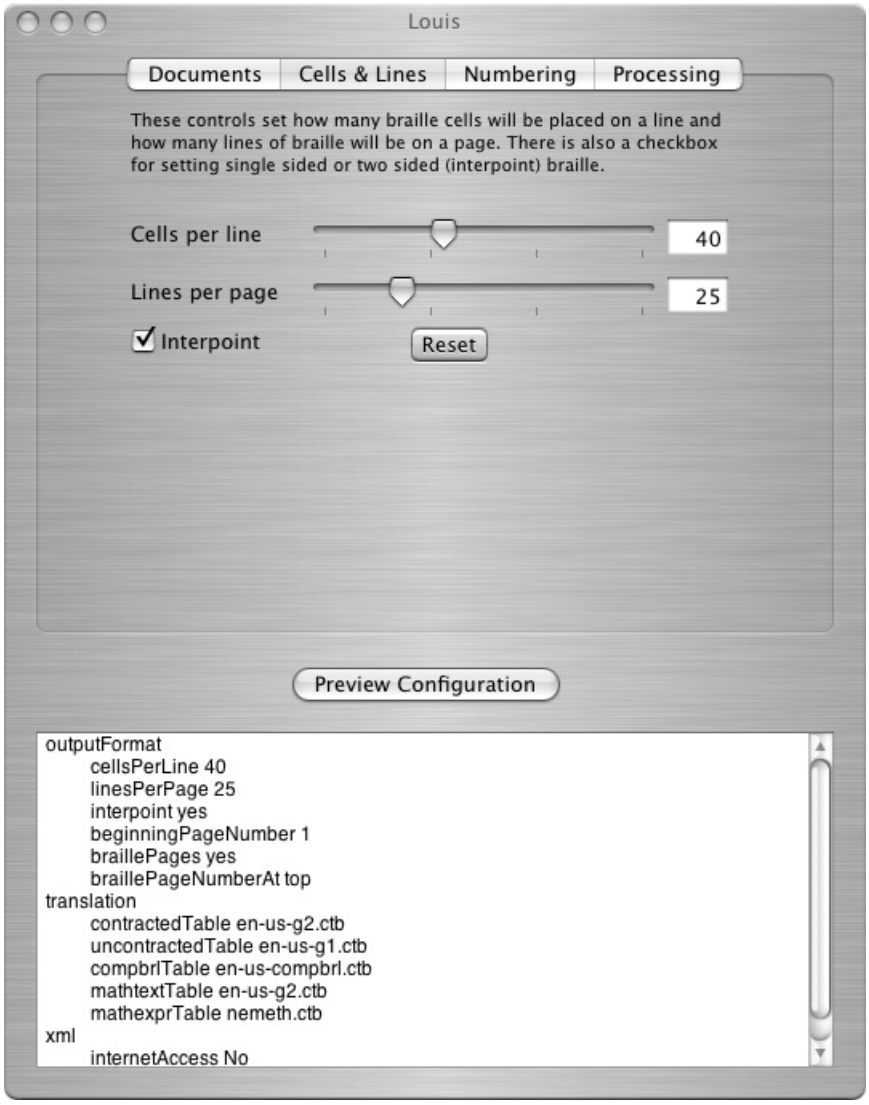

Figure 2: The Louis interface showing the Cells and Lines tab view.

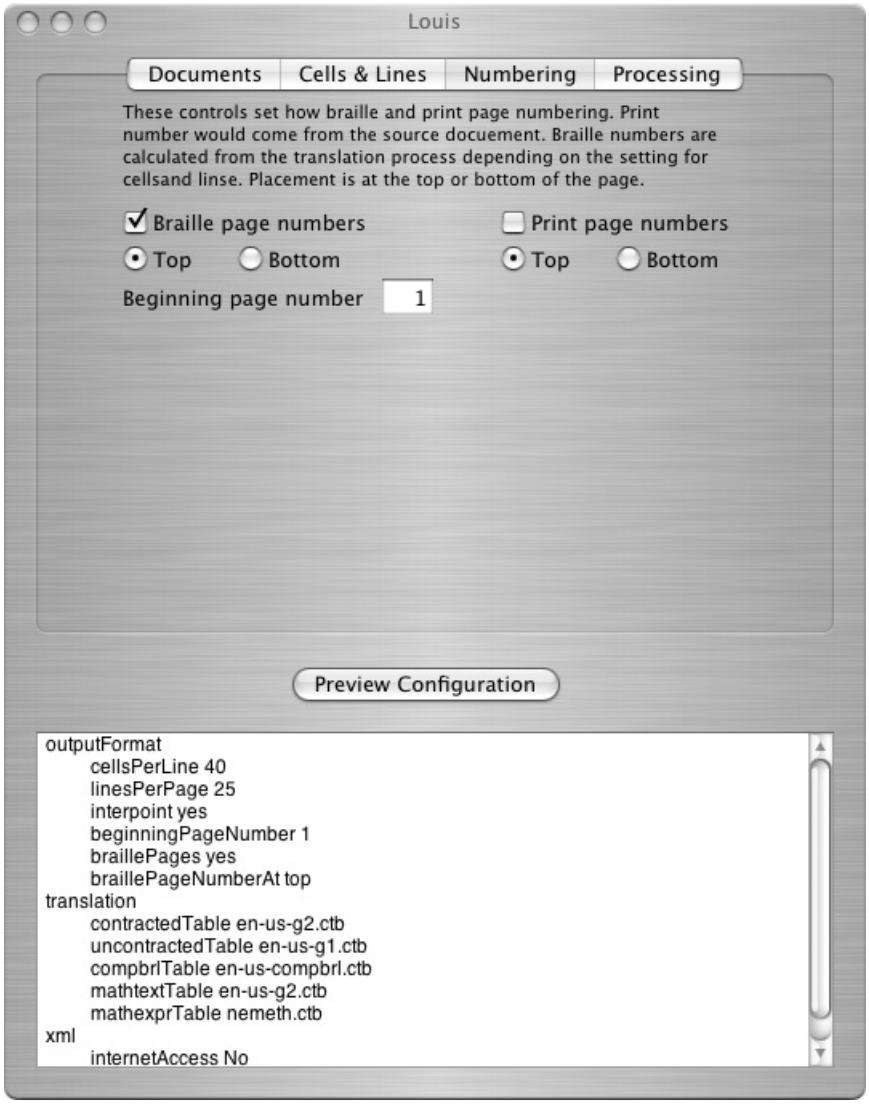

Figure 3: The Louis interface showing the Numbering tab view.

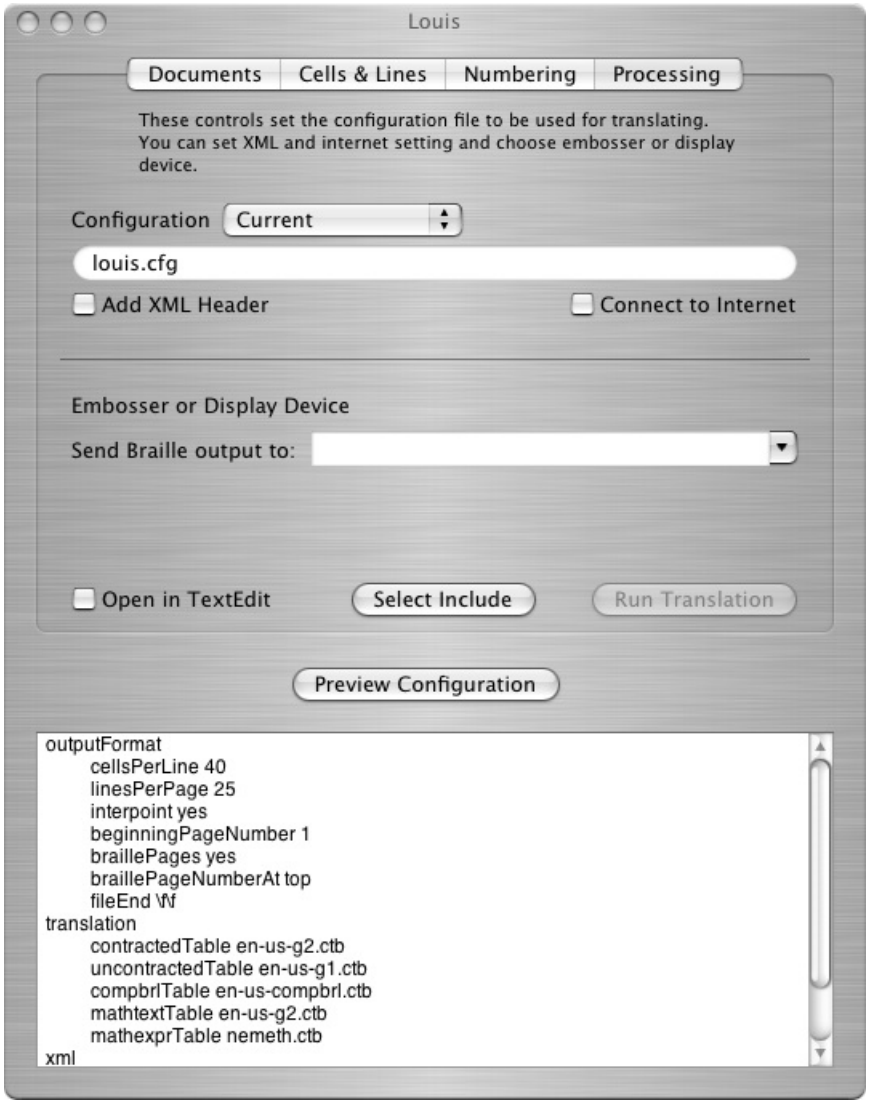

Figure 4: The Louis interface showing the Processing tab view.