User's guide to `xml2ly`

Jacques Menu

December 22, 2019 version

Abstract

This document presents the design principles behind `xml2ly`, as well as the way to use it. It is part of the `libmusicxml2` documentation, to be found at https://github.com/grame-cncm/libmusicxml/tree/lilypond/doc.

All the examples mentioned can be downloaded from https://github.com/grame-cncm/libmusicxml/tree/lilypond/files/samples/musicxml. They are grouped by subject in sub-directories, such as basic/HelloWorld.xml.

1 Acknowledgements

Many thanks to Dominique Fober, the designer and maintainer of the `libmusicxml2` library. This author would not have attempted to work on a MusicXML to LilyPond translator without it already available.

In particular, the conversion of MusicXML data to a tree is extremely well done directly from the MusicXML DTD, and that was a necessary step to produce LilyPond code. Dominique also provided a nice way to browse this tree with a two-shot visitor designe pattern, which this author used extensively in his own code. The interested reader can find information about that in libmusicxml2.pdf.

`xml2ly` and some of the specific examples presented in this document are this author's contribution to `libmusicxml2`.

2 Overview of `xml2ly`

2.1 Why `xml2ly`?

LilyPond comes with `musicxml2ly`, a translator of MusicXML files to LilyPond syntax, which has some limitations. Also, being written in Python, it is not in the main stream of the LilyPond development and maintainance group. The latter has much to do with C++ and Scheme code already.

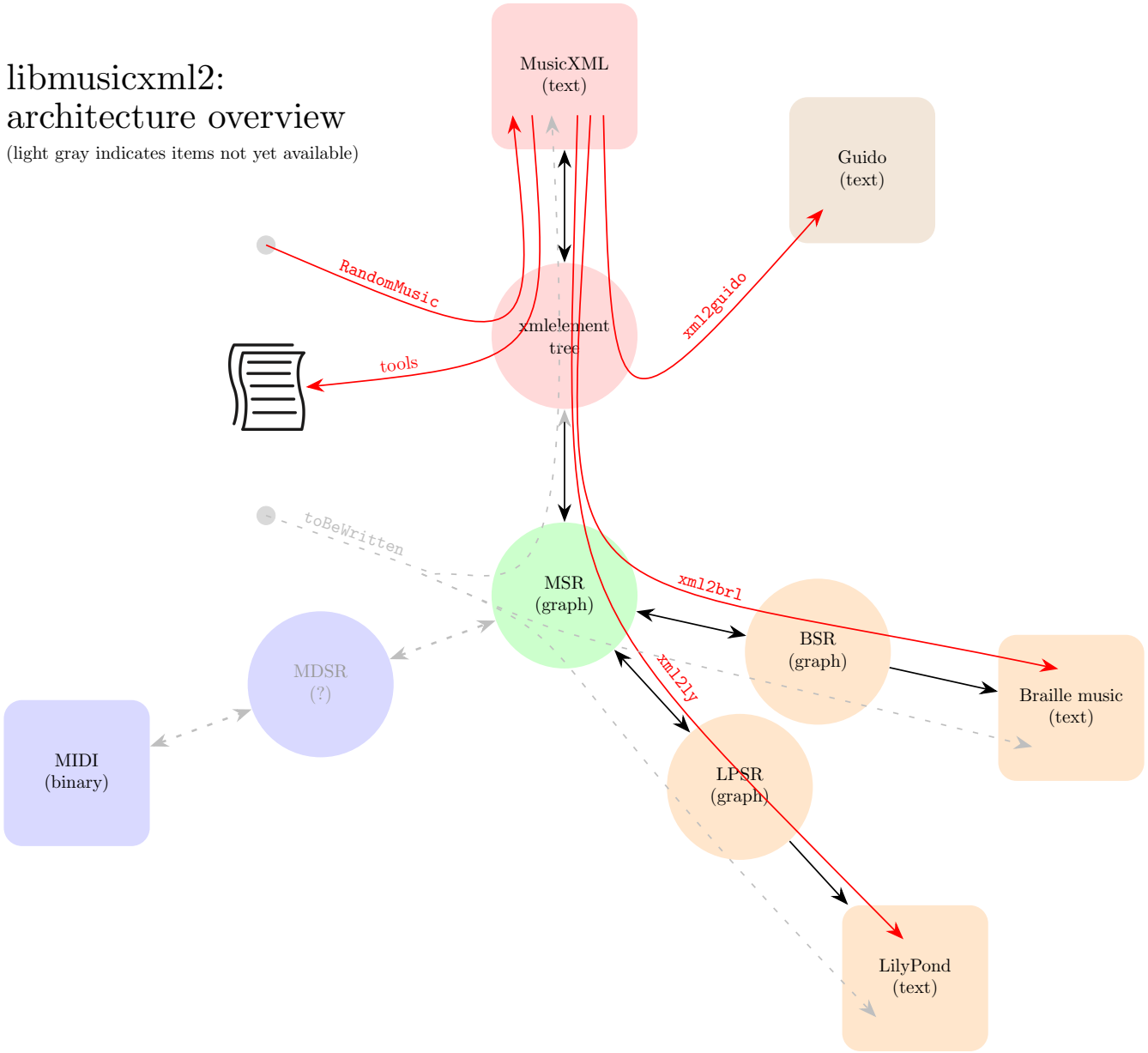
After looking at the `musicxml2ly` source code, and not being a Python developper, this author decided to go for a new translator written in C++.

The design goals for `xml2ly` were:

- to perform at least as well as `musicxml2ly`;
- to provide as many options as needed to adapt the LilyPond code generated to the user's needs.

Speed was not an explicit goal, but as it turns out, `xml2ly` is not bad in this respect.

Figure 1: `libmusicxml2` architecture



Entity	Description
xmlelement tree	a tree representing the MusicXML markups such as `<part-list>`, `<time>` and `<note>`
MSR	Music Score Representation, in terms of part groups, parts, staves, voices, notes, ...
LPSR	LilyPond Score Representation, i.e. MSR plus LilyPond-specific items such as `\score` blocks
BSR	Braille Score Representation, with pages, lines and 6-dots cells
MDSR	MIDI Score Representation, to be designed
`RandomMusic`	generates an xmlelement tree containing random music and writes it as MusicXML
tools	a set of other demo programs such as `countnotes`, `xmltranspose` and `partsummary`
`toBeWritten`	should generate an MSR containing some music and write it as MusicXML, LilyPond and Braille music
`xml2ly`	performs the 4 hops from MusicXML to LilyPond to translate the former into the latter
`xml2brl`	performs the 4 hops from MusicXML to Braille music to translate the former into the latter (draft)

- Note: `xml2ly` has a '`-jianpu`' option
- Note: `midi2ly` translates MIDI files to LilyPond code
- Note: `lilypond` can generate MIDI files from its input

`xml2guido` v2.3, `xml2ly` v0.9, `xml2brl` v0.01, August 2019

2.2 What xml2ly does

The architecture of libmusicxml2, which can also be seen at libmusicxmlArchitecture.pdf, is presented in figure 1. It shows the place of xml2ly in the whole.

The '-about' option to xml2ly details that somewhat:

```
menu@macbookprojm > xml2ly -about

What xml2ly does:

    This multi-pass translator basically performs 5 passes:
        Pass 1:  reads the contents of MusicXMLFile or stdin ('-')
                 and converts it to a MusicXML tree;
        Pass 2a: converts that MusicXML tree into to
                 a Music Score Representation (MSR) skeleton;
        Pass 2b: converts that tree and the skeleton into a
                 Music Score Representation (MSR);
        Pass 3:  converts the MSR into a
                 LilyPond Score Representation (LPSR);
        Pass 4:  converts the LPSR to LilyPond source code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    printing views of the internal data or printing a summary of the
   score.

    The activity log and warning/error messages go to standard error.
```

3 Options and help

xml2ly is equipped with a full-fledged set of options with the corresponding help. Since there are many options and the translation work is done in successive passes, the help is organized in a hierarchy of groups, each containing sub-groups of individual options called '*atoms*'.

3.1 Basic principles

Options are introduced on the command line either by '-' or '--', which can be used at will. There no difference between the two.

Each option has a short name and an optional long name. The latter is not needed if the short name is sufficiently explicit and not too long, such as '-jianpu', '-cubase', '-ambitus' or '-custos'.

Some options have their usual meaning in open-source software, such as '-h' (help), '-a' (about), and '-o' (output file name).

Some options name, short or long, share a common prefix, which allows them to be contracted, as in '-h=msr,lily', which is equivalent to '-msr, -lily', and '-trace=voices,notes', equivalent to '-trace-voices, -trace-notes'.

There are single-character options, which can be clustered: '-vac' is equivalent to: '-v, -a, -c'.

3.2 Introspection

One can obtain help on any specific group, sub-group or atom, such as:

```
menu@macbookprojm > xml2ly -option-name-help ambitus

--- Help for option 'ambitus' in subgroup "Engravers" of group "
    LilyPond" ---

LilyPond (-hlily, -help-lilypond):
  These lilypond control which LilyPond code is generated.
```

```
--------------------------
  Engravers (-hlpe, -help-lilypond-engravers):

    -ambitus
          Generate an ambitus range at the beginning of the staves/
    voices.
```

Some options have an optional value such as '`-option-name-help`', whose default value is... '`option-name-help`':

```
menu@macbookprojm > xml2ly -option-name-help

--- Help for option 'onh' in subgroup "Options help" of group "Options
    and help" ---

Options and help (-hoah, -help-options-and-help):
--------------------------
  Options help (-hoh, -help-options-help):

    -onh, -option-name-help[=OPTION_NAME]
          Print help about OPTION_NAME.
          OPTION_NAME is optional, and the default value is 'onh'.
```

3.3 Trace options

`xml2ly` is equipped with a range of trace options, that are crucially needed by this author when testing and fine-tuning the code base.

The bulk of these options is placed in a group that is hidden by default:

```
  Trace (-ht, -help-trace) (hidden by default)
  --------------------------
```

The interested reader can see them with the '`-help-trace`' group option:

```
menu@macbookprojm > xml2ly -help=trace

--- Help for group "Trace" ---

Trace (-ht, -help-trace) (hidden by default)
  There are trace options transversal to the successive passes,
  showing what's going on in the various translation activities.
  They're provided as a help to the maintainers, as well as for the
    curious.
  The options in this group can be quite verbose, use them with small
    input data!
  All of them imply '-tpasses, -trace-passes'.
--------------------------
  Options handling trace          (-htoh, -help-trace-options-handling)
    :
    -toah, -trace-oah
          Write a trace of options and help handling to standard error.
          This option should best appear first.
    -toahd, -trace-oah-details
          Write a trace of options and help handling with more details
    to standard error.
          This option should best appear first.
  Score to voices                 (-htstv, -help-trace-score-to-voices)
    :
    -t<SHORT_NAME>, -trace<LONG_NAME>
          Trace SHORT_NAME/LONG_NAME in score to voices.
          The 9 known SHORT_NAMEs are:
            score, pgroups, pgroupsd, parts, staves, st, schanges,
    voices and voicesd.
```

```
          The 9 known LONG_NAMEs are:
            -score, -part-groups, -part-groups-details, -parts, -staves
    .
... ... ... ... ... ...
```

As can be seen, there are event options to trace the handling of options and help by `xml2ly`. The source code contains many instances of trace code, such as:

```
#ifdef TRACE_OAH
  if (gTraceOah->fTraceVoices) {
    gLogOstream <<
      "Creating voice \"" << asString () << "\"" <<
      endl;
  }
#endif
```

Building `xml2ly` with tracing disabled only gains less than 5% in speed, this is why tracing is available by default.

3.4 Non-musical options

3.4.1 Timing measures

There is a '`-cpu`' option to see show much time is spent in the various translation activities:

```
menu@macbookprojm > xml2ly -option-name-help cpu

--- Help for option 'cpu' in subgroup "CPU usage" of group "General"
    ---

General (-hg, -help-general):
----------------------------
  CPU usage (-hgcpu, -help-general-cpu-usage):

    -cpu, -display-cpu-usage
          Write information about CPU usage to standard error.
```

In practise, most of the time is spent in passes 1 and 2b. The '`time`' command is used to obtain the total run time, since `xml2ly` cannot account for input/output activities:

```
menu@macbookprojm > time xml2ly -aofn -cpu xmlsamples3.1/
    ActorPreludeSample.xml
*** MusicXML warning *** xmlsamples3.1/ActorPreludeSample.xml:44: <
    system-distance /> is not supported yet by xml2ly
... ... ... ... ...
*** MusicXML warning *** xmlsamples3.1/ActorPreludeSample.xml:27761: <
    direction/> contains 2 <words/> markups
Warning message(s) were issued for input lines 44, 45, 46, 551, 584,
    732, 1121, 1215, 4724, 27761

Timing information:

Activity                        Description       Kind  CPU (sec)
--------  --------------------------------  ---------  ---------

Pass 1    build xmlelement tree from file  mandatory  0.268994
Pass 2a   build the MSR skeleton           mandatory  0.076413
Pass 2b   build the MSR                    mandatory  0.276732
Pass 3    translate MSR to LPSR            mandatory  0.056381
Pass 4    translate LPSR to LilyPond       mandatory  0.082213

Total      Mandatory  Optional
-------    ---------  ---------
0.760733    0.760733   0
```

```
21
22
23 real  0m0.814s
24 user  0m0.751s
25 sys 0m0.058s
```

This compares favorably with `musicxml2ly` measurements:

```
1 menu@macbookprojm > time musicxml2ly xmlsamples3.1/ActorPreludeSample.
     xml
2 musicxml2ly: Reading MusicXML from xmlsamples3.1/ActorPreludeSample.xml
     ...
3 musicxml2ly: Converting to LilyPond expressions...
4 ... ... ... ... ...
5 musicxml2ly: Converting to LilyPond expressions...
6 musicxml2ly: Output to 'ActorPreludeSample.ly'
7 musicxml2ly: Converting to current version (2.19.83) notations ...
8
9 real  0m4.113s
10 user  0m3.659s
11 sys 0m0.407s
```

3.4.2 Chords informations

Listings

List of Figures

Contents