Lilypond for pyScore: Approaching a universal translator for music notation

Stephen Sinclair
Schulich School of Music
McGill University
Montreal, Quebec
sinclair@music.mcgill.ca

Michael Droettboom
formerly of Johns Hopkins University
3400 North Charles Street
Baltimore, Maryland, USA 21218
mike@droettboom.com

Ichiro Fujinaga
Schulich School of Music
McGill University
Montreal, Quebec
ich@music.mcgill.ca

Abstract

Several languages for music notation have been defined in recent years. pyScore, a framework for translating between notation formats, and new module for it which can generate input for the LilyPond music engraving system are described. This shows the potential for developing pyScore into a "universal translator" for musical scores.

Keywords: Notation, score, engraving, translation, representation.

1. Introduction

Recent years have seen a large increase in the use of computers for representing musical notation. This is due to an increased interest in digital archiving of musical scores, computer typesetting, score editing, optical music recognition, web-based distribution of music, and interchange between the variety of available software programs.

Two forerunners have most recently been touted as solutions for distribution and interchange. These are Recordare's MusicXML [3], an "Internet-friendly" XML-based format, and GUIDO, a "representationally adequate" and "human readable" text format [4]. They are both designed and built on previous work, including, but not limited to, the binary NIFF format, SMDL, and Humdrum's **kern [5].

Another contender is a free and open-source music engraving system called LilyPond [6]. Its input format is intended more as a description of visual layout than for data interchange, but due to the professional quality of its output, some have begun to use its input format to store musical data.

One reason for the encouraging support it has obtained is that its input format is similar in many ways to the popular LaTeX document typesetting software. The grammar is highly flexible and quite easy to read and write. For example, musical expressions can be defined as "commands," to be later be recalled in one or more contexts. Different instrument voices can then be defined separately and specified in a more specific layout context later in the document. Also, any variable used for control of its Scheme-based layout system can be overridden from within the input script. Finally, many special cases are supported for instruments with exceptional notation, such as bagpipes or percussion, as well as ancient notation and special constructs for contemporary music.

Because it is free and very extensible, it can be used as a good solution for displaying results from MIR transactions. An easy way to generate its input format for automated systems would be highly desirable. In this paper, we discuss a software framework for translating musical scores. In addition to its previous support for MusicXML and GUIDO, we have created a new module for generating input to LilyPond.

2. Translating notation with pyScore

Because of the multiplicity of formats that exist, it is impossible to expect every software package to know how to parse them all. Instead, it is important to be able to independently translate foreign formats to familiar ones, bridging otherwise incompatible programs. Additionally, storing musical information in open formats ensures data longevity. Translation can allow for this, while ensuring compatibility with a wide array of software tools.

While many programs exist for specific translations between particular formats, a more global approach to this problem is taken by pyScore, a framework developed in the Python programming language [1]. Currently, it can work with MusicXML and GUIDO. It is able to read them into internal tree structures which can then be translated from one to the other, or into MidiXML, which can in turn output MIDI data as a file or a stream. It can also render GUIDO to image files by calling the online NoteServer.

The framework is structured so that it is quite simple to add support for new formats. When a module is added, pyScore is able to find a path from whatever input format is given to the new module. Thus, given the ability to translate a GUIDO tree into a LilyPond tree, it is automatically able to also perform the internal conversion from MusicXML to GUIDO if necessary. This feature, in addition to the clear and modular programming framework, and combined with the advantages of using Python (such as cross-platform

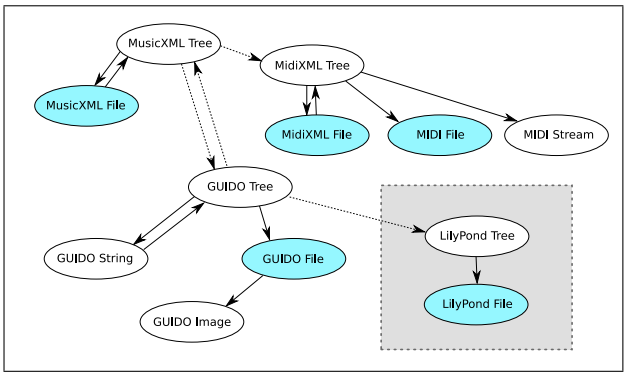

Figure 1. Conversion graph for pyScore, with LilyPond module added.

compatibility, and readability of the source), make pyScore a particularly well-adapted platform for building a universal score converter.

We have created a LilyPond module for pyScore. The new conversion graph is shown in Figure 1. Currently it supports conversion *to* LilyPond. It can handle the complete definition of the "Basic" subset of GUIDO, meaning that it supports most important musical features, but currently lacks much of the layout information that is possible to specify using GUIDO's "Advanced" counterpart.

3. Representational differences

There are inevitably some choices that must be made when formalizing an abstract concept like music. Some of these choices are human considerations, to increase clarity and to ease manual editing, while others make fundamental differences in how the computer must make assumptions or numerically handle information. Different systems will invariably choose different ways of encoding structural relationships that exist within music at every level of abstraction [2].

As a short example, consider the seemingly simple construct of a tuplet. In a triplet, each note has a duration of $\frac{2}{3}$ of its normal duration. GUIDO represents this by specifying each note with its absolute duration, multiplying by a fraction if necessary:

```
[c*1/3 c*1/3 c*1/3]   % Half-notes, 1 bar
[c*1/6 c*1/6 c*1/6]   % Quarter-notes, 1/2 bars
```

This is representationally simple, but does not explicitly define the relationship between the notes. In contrast, LilyPond chooses to specify normal durations, and multiply the group of notes by a fraction:

```
\times 2/3 { c'2 c' c' }   % Half-notes, 1 bar
\times 2/3 { c'4 c' c' }   % Quarter-notes, 1/2 bars
```

This difference implies that when reading GUIDO, the translator must detect groups of notes to be considered tuplets, since they are not already grouped in the input file. This is done by checking each note duration to see if it is an even power of two, and if not, collecting them until the total duration of the group matches this criteria.

Design choices such as these make it often impossible to derive a universal solution for translating symbolic information. pyScore approaches this by performing specific translations from one language to another in series, rather than trying to define a one-size-fits-all internal representation. While long chains of translations may be lossier than necessary, this means individual translations can be optimized to reduce loss of information.

4. Conclusion

pyScore is a useful framework for building a notation translator. By using it to convert from GUIDO to LilyPond, we were able to take advantage of a ready-made GUIDO parser. As a bonus, the software is automatically able to translate MusicXML into LilyPond, by discovering the appropriate intermediate conversion. This allowed us to concentrate solely on LilyPond's representation, without having to track differences between three grammars.

Future work will support more of the Advanced GUIDO features. Additionally, a parser for the LilyPond format should be added, to enable conversion *from* LilyPond. This may present a more difficult task, because of the flexibility of the LilyPond language, and its escape mechanism for the software's internal Scheme interpreter.

It would be beneficial to add yet more format support to pyScore, to eventually achieve an open-source "universal translator" for music.

5. Acknowledgments

We are grateful to National Science Foundation, Institute for Museum and Library Services, the Lester S. Levy Family, and Canadian Foundation for Innovation for their financial support.

References

[1] pyScore [Software], 2006 Apr 24; available from: `http://pyscore.sf.net/`.

[2] R. Dannenberg, "Music representation: Issues, techniques, and systems," in *Computer Music Journal*, vol. 17, no. 3, pp. 20–30, 1993.

[3] M. Good and G. Actor, "Using MusicXML for File Interchange," in *Proceedings Third International Conference on WEB Delivering of Music*, 2003, p. 153.

[4] H. Hoos, K. Hamel, and K. Renz, "Using Advanced GUIDO as a Notation Interchange Format," in *Proceedings of the International Computer Music Conference*, 1999.

[5] D. Huron, "Music information processing using the Humdrum toolkit: Concepts, examples, and lessons," in *Computer Music Journal*, vol. 26, no. 2, pp. 11–26, 2002.

[6] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, 2003.