choice

Encoding and Visualizing Variants in LilyPond

Urs Liska
ul@openlilylib.org
openLilyLib
University of Music Freiburg

One area of significant progress in the LilyPond ecosystem over the past five years has been scholarly applications, such as encoding annotations, producing critical reports, adding analytical markup, or integrating text and music. This paper presents a set of relatively new packages or modules from the *openLilyLib*[1] extension framework which promise to drive development of LilyPond as a scholarly tool forward even more: `scholarly.choice` allowing the encoding of multiple variants, `scholarly.editorial-markup` extending the concept of editorial annotations beyond the existing `scholarly.annotate`, and `stylesheets.span` as the underlying encoding basis and the foundation of a planned comprehensive stylesheet framework.

The module `scholarly.annotate` has been around for some years and provides tools for encoding scholarly annotations within the LilyPond input file. Annotated items can be highlighted with colors, and annotations can be exported to various file types, including LaTeX files to be used as source for high-quality typesetting of critical reports. It also includes basic support for visual highlighting of editorial decisions based on score type, but this had not been properly fleshed out so far. But the more important conceptual limitation of the `annotate` module is that annotations essentially amount to *comments* where the documentation of the source – if relevant to the editorial purpose – has to be done within the annotation *text*. There was a demand for a tool allowing to encode both the original reading from the source *and* the edited text, along with an annotation. As a response to this demand a comprehensive solution is being developed which draws heavily from the “Critical Apparatus” and ”Editorial Markup” sections of the MEI specification.

The underlying functionality is implemented in a module `span`, which is inspired by the HTML element of the same name. Essentially this is used to semantically encode some music “as something” and give access to various highlighting scenarios. By default simply a color is applied to the spanned music, depending on the span’s “class”, but functions can be provided to style the music according to the span class *and* the score element’s type (e.g. a span class “foo” might dash slurs, parenthesize accidentals, and print note heads at a smaller size). Right now control of the styling requires programming Scheme functions, but the goal is to eventually provide a sort of meta language allowing to do the styling of a musical score in a similar way to styling a website with CSS. In addition

[1]`https://openlilylib.org`, development: `https://github.com/openlilylib`

to the actual styling it is possible to use span properties to generate additional markup such as footnotes, text annotations, music examples, or temporary ossia staves.

`scholarly.editorial-markup` builds upon the `span` module, tailoring its use to scholarly applications by providing preset span classes (modeled after MEI elements) along with rules governing expected, required or allowed properties: apart from generic, MEI-inspired properties like "certainty" or "reason" there are more specific cases such as for example `lemma` or `reading` which require a `source` attribute.

Finally, the `scholarly.choice` module provides a clean interface to encode arbitrary differing musical texts in parallel, for example readings from different sources, modifications within a source, or editorial emendations. Depending on the edition's purpose these encodings can be used merely for documenting alternative readings, or to selectively render different versions of the score. Critical annotations can be carried along with the encoding, applicable either on the level of the overall choice or at the span that is chosen to be rendered. Since this is ultimately based on the `span` module all the highlighting options described above are available for `choice` as well.

The paper describes both the tools and the development, giving more details about the conceptual background, current possibilities, and further plans. Examples will be taken from the DME's new digital edition of Leopold Mozart's *Violin School* where the `choice` module is currently being used to encode variants in the book's different editions within single source files. The encoding aspect is particularly relevant in this edition because the LilyPond input files are directly included in the edition's TEI files, which imposes strict requirements on the code's readability and semantic clarity.

One additional module may or may not be completed by the time of the MEC (and will or will not be discussed in the paper accordingly) is `scholarly.sources` that can be used to encode information about relevant sources. Planned functionality includes typesetting source descriptions and maintaining inheritance information to eventually draw stemmas for use in critical reports.