Abjad, A Python API for Formalized Score Control

Trevor Bača
Harvard
trevorbaca@gmail.com

Josiah Wolf Oberholtzer
Harvard
josiah.oberholtzer@gmail.com

Jeffrey Treviño
Carleton College
jeffrey.trevino@gmail.com

Victor Adán
Bank of America
vctradn@gmail.com

ABSTRACT

Place your abstract at the top left column on the first page. Please write about 150-200 words that specifically highlight the purpose of your work, its context, and provide a brief synopsis of your results. Avoid equations in this part.

1. INTRODUCTION

2. BACKGROUND & HISTORY

3. NOTATIONAL ISOMORPHISM

Abjad models objects on the page according to common practice notation.

One class per user-creatable notational element: Note, Chord, Rest, Articulation, Slur, Beam, Tie.

3.1 Explicit notational modeling

Abjad models notation explicitly.

Durations: written, assignable.

`Notes`, `Chords` and `Rests` must be instantiated with assignable written durations. Durational assignability tests whether a duration can be represented as a power-of-two flag count combined with zero or more dots. `7/16` is an assignable duration while `5/32` and `9/8` are not. Non-assignable durations cannot be represented in common practice notation by a single glyph but only by two or more glyphs tied together. Abjad will not automatically render a single note with a duration of `5/16` as two or more notes tied together. We consider such behavior to be too implicit. there are too many potentially compositionally valid ways to render a duration such as `5/16` into a series of tied assignable durations: `1/4 + 1/16`, `3/16 + 2/16`, `2/16 + 3/16`, `1/16 + 1/4`, `1/8 + 1/8 + 1/16` etc.

Instead we provide affordances for generating tied notes from non-assignable durations via the `scoretools.make_notes()` function.

```
>>> notes = scoretools.make_notes(["c'"], [(5, 16)])
>>> staff = Staff(notes)
>>> print(format(staff))
```



```
\new Staff {
    c'4 ~
    c'16
}
```

Prolation: diminution vs augmentation.

Likewise, `Notes` and other leaves cannot be instantiated with durations with non-power-of-two denominators. Not only are such durations non-assignable but they also suggest implicit tuplet-derived prolation..

Three notes each having a prolated duration of `1/12` can be represented as either three `1/16` notes in a `3:4` tuplet or three `1/8` notes in a `3:2` tuplet.

3.2 Notational aggregation

We assume notational primitives are the elements of composition.

The act of composition revolves around the aggregation of these primitives into arbitrarily complex score objects.

Examples: append(), extend(), insert(), attach().

3.3 Notational visualization

Abjad makes visualizing notational artifacts simple. Any notational element or aggregate can be displayed at any time as a PDF via calls to its top-level `show()` function.

4. RELATIONSHIP MODELING

5. SCORE ADDRESSABILITY

6. EXTENSIBILITY

7. EMBEDDABILITY

Abjad is an importable Python library. It can be used in whole or in part as a component of any Python-compatible system. Abjad has few Python package dependencies and is not bound to any specific user application or graphic user interface. These qualities make Abjad an ideal project ideal for embedding in other software systems.

For example, Abjad supports IPython Notebook [1] , a web-based interactive computational environment combining code execution, text, mathematics, plots and rich media into a single document. Notational output from Abjad can be transparently captured and embedded directly into an IPython Notebook which has loaded Abjad's IPython Notebook extension.

[1] http://ipython.org/notebook.html

8. OPEN SOURCE

Acknowledgments

You may acknowledge people, projects, funding agencies, etc. which can be included after the second-level heading "Acknowledgments" (with no numbering).