Charles Winston
charles.winston@tufts.edu

Improving the Internal Structure of Chords

Summary

LilyPond's current internal representation of chords is not powerful enough to capture many meaningful musical characteristics. A chord is a set of simultaneous pitches. These pitches are precisely the notes which make up the chord—the notes you hear and the notes you see on the staff. But what makes up a chord are not solely the pitches, but other characteristics valuable in a music theory context. These semantics of a chord move beyond the notes themselves and deal with the musical context of the chord. Examples of such characteristics are: the root, the quality, the scale degree, etc. The goal of this project is to properly define these semantics and include them in the internal representation of chords.

I will build upon the current music object representing chords, EventChord, so it can now handle both these musical semantics as well as simple note events of the chord. The structure must have three avenues. (1) It must have an entry which supports the individual notes of the chord. This structure is already in place with EventChord's `elements` property, which is a list of NoteEvents. (2) It must keep its current `articulations` property, which holds any articulations for the chord. (3) There must be a new entry added to the structure: a `semantics` property. This semantics entry will include many fields. The fields that I deem semantically necessary are:

- root
- quality (major, minor, diminished, etc.)
- extensions (2nd, 7th, 9th, 13th etc.)

Other fields will be added, of which may include:

- scale degree
- voicing/inversion
- bass note

The complete list of aspects will be finalized during community bonding after much discussion with both the user and development community. And of course, this structure should allow for other semantic features to be easily added in the future.

Benefits

Modifying the internal representation of chords will result in many great benefits. This new representation now has three useful avenues, each of which can be used independently in other contexts and by other engravers. For example, inside the ChordNames context, the `Chord_name_engraver` can now accept a new `chord-semantics` type, having the option to ignore the individual note events of the chord and focus only on semantics in order to produce a name. The result is a simpler and more powerful process of chord naming, being able to name more complex chords, and do so more accurately.

Further, by having an internal representation of chords that deals with musical semantics, a new door is opened on the other side of which lies an array of new possibile functions. For example, one possibility could be displaying chords not by name, but by their scale degree (i.e. I, ii, iii, etc.). When we have a representation that captures more than just notes, but the musical *context* of chords, we have the ability to perform many more important musical functions.

Deliverables

As said above, a new entry must be added to EventChord which captures the semantic value of the chord. This will be a new property called `semantics`, which will be defined in `define-music-properties.scm`. The definition of the EventChord object is contained in the file `define-music-types.scm`, and the description will be updated to contain this third, new property. A new type `chord-semantics` must also be defined.

After adding these new features to EventChord, other modules which use EventChord must be updated as well. These are mainly:

- Chord entry: the file `chord-entry.scm` creates an EventChord, and must be modified appropriately.
- Chord naming: the ChordNames context properties can be updated in the file `define-context-properties.scm`. We must also update the `Chord_name_engraver` to accept the `chord-semantics` type.

- Input and output may also need to be modified in the future to better suit this representation. This is not necessarily in the scope of this project, but is important to note.

Plan

The first action that will take place is planning and designing the new property, `semantics`. When implemented, it will be added to the properties of EventChord. The midterm goal will be to complete the implementation of the new features EventChord, and update `chord-entry.scm` to create EventChords based on this new representation. The implementation can be tested by the `\displayMusic` option which displays the internal representation of a music expression.

After this is completed, ChordNames will be updated to accept the new type `chord-semantics` and produce names based on the `semantics` property.

Smaller deadlines before the midterm completion will of course be established. Much of the planning will take place during community bonding, where the community will discuss which semantic features are desired for this new representation.

Documentation will be written during the progress of the project. Documentation for this project is very important and must be done well, because we are changing important structural aspects of the representation of chords.

Communication

My mentor Carl Sorenson and I will be able to effectively communicate via email. I anticipate there being some times where a Skype chat may be beneficial, and so we will do that as well. GitHub will be our primary mode of code sharing, and commits will be made often so as to keep up a frequent revision process. Also, lilypond-user and lilypond-devel will be great and important resources, and will inform many of the decisions I make, especially in planning and defining what exactly should be included in the semantics of a chord.

Qualification

I am a musician and a programmer, and it's hard to say which one I am first. This project appealed to me due to my interest in music theory. I think about chords in terms of these semantics we are defining. For me, chords always have specific roles within keys and within pieces of music, and I would love to be able to build a representation that captures these important characteristics.

I am incredibly excited to work on a project of this size and scope, and see this as a wonderful opportunity to marry two of my great passions. I have worked in Max and so I have some general programming experience in a music context, but I have never worked on building music software before, and I have also not worked on free software in general. Pursuing this project excites me a great deal and is something that I will invest a lot of time and hard work into.

Of the skills that I would need to complete the project, the only ones I will have to learn are specific programming languages (Scheme, which I have already learned a great deal of). I have great knowledge of music theory and chord naming, and I have much experience in object oriented programming, implementing complex structures that encapsulate information which will lend itself well to building the chord representation.