G♭: Language Specification

Ezra Joffe-Hancock and Zach Stein-Perlman

Spring 2022

1 Introduction

Ezra plays alto sax in a jazz combo here, and when he's struggling with soloing over certain songs, it helps him to run over the chord changes and play along with the recording, but the best is when there's a backing track on YouTube, so he can play along with the chords without drowning whoever is soloing on the recording.
We created a language that makes it easy to quickly create a backing track to play over. The programmer creates sections of music by arranging chords and choosing their durations, then describes how many times and in what order to play each section. The output is a corresponding audio file, or at least an xml file that describes audio.

2 Design Principles

The language is simple and accessible for a non-technical user. The syntax is natural. The order of commands makes programs easily readable with intuitive layout. Simplicity is prioritized (worse is better!). A valid program is ideally easy to read and facilitates quickly understanding the music that it describes.

3 Example Programs

Example 1:

```
let Chorus = {
Cmaj 1
Fmaj 1
Cmaj 1
Gmin 1
Cmaj 4
}

play Chorus 2
```

Example 2:

```
let Chorus = {
Emaj 2
G#maj 2
Bmaj 2
Emaj 4
}

play Chorus 2
```

Example 3:

```
let Intro = {
Cmaj 2
}

let Chorus = {
Gmaj 1
Gmin 1
}

play Intro 1 Chorus 10 Intro 1
```

4 Language Concepts

In order to program in the language, a user should understand the sounds/names of different instruments, have a basic understanding of chords (so they can specify them by writing out their letters), an understanding of a bar/measure which is a grouping of several beats of music. They should understand also how to transfer information from a written piece of sheet music with chords to information within the language, by copying down chords and putting them loops for the appropriate bars. We hope to also provide specifiers like volume, and tempo, so a user should understand how those qualities would affect the output.

5 Syntax

Programs have two parts: defining sections and playing them. When defining a section, we give the name of the section and the chords it comprises. To play, we use its name and the number of times we want to play it.

```
<expr>        ::= <assignments> <play>
<assignments>::= <assignment>+
<assignment> ::= <section> <variable>
<section>     ::= <sound>+
<sound>       ::= <chord> <ws> <int>
<chord>       ::= <note> <ws> <quality>
<ws>          ::= ␣
<note>        ::= A | A# | Bb | B | C | C# | Db | D | D# | Eb | E | F | F#
                | Gb | G | G# | Ab
<quality>     ::= Maj | Min
<variable>    ::= <string>
<play>        ::= (<variable> <int>)+
```

6 Semantics

i What are the primitive kinds of values in your system? One kind of primitive value in our language are notes, which are of type char and are later converted using some sort of dictionary structure into either floats representing note frequencies, or directly to audio samples.

  Another primitive are integers, which are used to define the values of volume, tempo, bars range, and number of repeats of sections. Chords are a primitive and are a list of notes, and are created by interpreting a the name of a chord into its component notes according to some function that operates according to the rules of music theory.

Syntax	Abstract syntax	Type	Prec./Assoc.	Meaning
F#	Note	string	n/a	A note
Cmaj	Quality of Note	string → Chord	n/a	A chord to be played where C stands for the note and maj for the quality
Cmaj 1	Sound of Chord * int	(Chord * int) → Sound	n/a	A chord and its duration
sound1 ... soundn	Section of Sound list	Sound list → Section	n/a	A sequence of sounds
Chorus	Variable of string	string	n/a	The name of a section
let var = {section}	Assignment of Section * Variable	(Section * Variable) → Assignment	n/a	A pairing of a section with a name
assign1 ... assignn	Assignments of Assignment list	Assignment list → Assignments	n/a	A list of sections and their associated names
play var1 x1 ... varn xn	Play of (Variable * int) list	(Variable * int) list → Play	n/a	A specification of sections to play
Assignments Play	Program of Assignments * Play	Assignments * Play → Program	n/a	A complete program

ii What are the "actions" or compositional elements of your language? In other words, how are values combined?

Sections are marked by a "let sectionName = {" command and then a series of sounds until the "}" command is reached. These sections are then given as input to the play function, which produces the output of a section (a combination of sounds) a given number of times.

iii A Our programs read no input.

B Our programs output an xml describing audio

7 Remaining Work

Necessary data types: all implemented.

Necessary operations: all implemented.

We will add more chord qualities.

We will add slightly more control over tempo.

Possible stretch goals:

- Program correctness checker, including at least case-insensitivity
- Enhance readability of language specification
- Add instruments beyond piano
- Allow more flexible control over tempo, volume, and/or octave
- Pretty printing of musical notation corresponding to the audio