Solver-Aided Chorale Composition

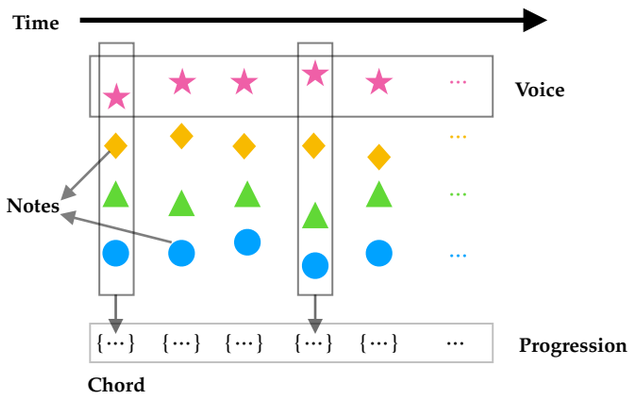
Junrui Liu

1 Introduction

Composing music is hard: if you hit the keys on a piano randomly, chances are, it won't sound great. A music theory aims to distill a musical *syntax* such that a syntactically correct composition will not sound "wrong." Thus, composition can be viewed as a constraint satisfaction problem: given a background theory, find a sequence of composition actions that does not violate the theory's syntax. We corroborate this view by developing a tool called *Choco* for composing music in the style of Baroque chorales using Rosette [1]. This project is inspired by [2].

2 Background

A chorale typically consists of four independent *voices*. Each voice is a sequence of *notes* drawn from a fixed collection known as a *scale*. At each time step, the set of notes from all voices form a *chord*, whose evolution is called a *progression*.



References

[1]: Torlak, Bodik. A Lightweight Symbolic Virtual Machine for Solver-Aided Host Languages. PLDI'14.

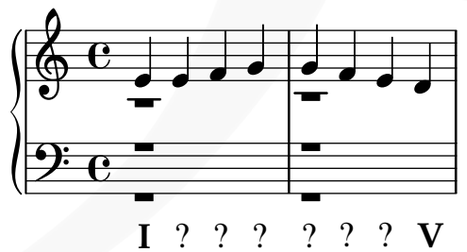
[2]: Cong, Leo. Counterpoint by Construction. FARM'19.

[3]: (Similar work) https://github.com/kach/recreational-rosette/tree/master/music.

3 Overview

(i) The user provides a *chorale sketch*, which may contain *symbolic notes* and *symbolic chords*.
In the sketch on the right:

- The top voice is concrete, while all remaining voices are symbolic, to be filled by the solver.
- The progression's first and last chords are specified (I and V), while all remaining chords undetermined.



(ii) Our framework encodes the syntactic rules of classical harmony as logical predicates. E.g.,

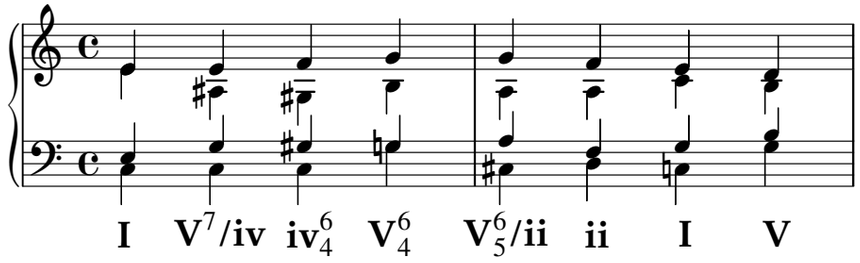
- All voices are in harmony w.r.t. the chord.
- Voices cannot "cross" each other.

$\forall n \in \mathscr{V} \,.\, \exists pc \in \mathscr{C} \,.\, n \equiv pc$

$\forall n, m \in \mathscr{V}_t \times \mathscr{V}_{t+1} \,.\, \mathsf{V}(n) < \mathsf{V}(m) \implies n < m$

(iii) We use angelic execution to fill the sketch in a way that satisfies all syntax rules.

```
(define model (solve (assert cs)))
(evaluate sketch model)
```



4 Optimizations

1. Tabulate expensive symbolic computation via pre-computation.
2. Modular synthesis via horizontal (temporal) and vertical (chordal) decomposition wth backtracking.
3. Transform constraints to avoid expensive modular operations.

5 Future Work

1. Design a DSL for specifying syntax of different music genres and theories.
2. Frame composition as optimal synthesis problem to account for soft syntactical constraints.
3. Extend framework to incorporate metric theory and transformational theory.