Judah Daniels

Inferring Harmony from Free Polyphony

Computer Science Tripos – Part II

Clare College

July, 2023

Declaration of originality

I, Judah Daniels of Clare College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose. I am content for my dissertation to be made available to the students and staff of the University.

Signed Judah Daniels

Date April 5, 2023

Proforma

Candidate Number:	**2200D**
Project Title:	**Inferring Harmony from Free Polyphony**
Examination:	**Computer Science Tripos – Part II, July, 2023**
Word Count:	**2684**[1]
Code Line Count:	**2272**
Project Originator:	**Christoph Finkensiep**
Supervisor:	**Dr Peter Harrison**

Original Aims of the Project

Work Completed

All that has been completed appears in this dissertation.

Special Difficulties

None

[1]This word count was computed by `detex diss.tex | tr -cd '0-9A-Za-z \n' | wc -w`

Contents

List of Figures

Acknowledgements

Chapter 1

Introduction

This dissertation explores efficient search strategies for parsing symbolic music data using a musical grammar for Automatic Chord Estimation (ACE). We first present a naive implementation of a parsing algorithm based on a recent grammatical model, then address problems of intractability through classical heuristic search methods. We will see that that my novel heuristic search algorithm achieves commendable results, providing a strong foundation for a more sophisticated automated analysis system.

1.1 Motivation

Most of western tonal music can be described using a sequence of chords, representing a higher level harmonic structure of a piece. Automatic Chord Estimation (ACE) is the task of inferring the sequence of chords for a given piece from symbolic (such as a score) or audio data. There is a small, finite set of chord types, but each chord can be realised on the musical surface in a practically infinite number of ways. Given a score (a symbolic representation of a piece of music), we wish to infer the sequence of underlying chord types.

Automatic Chord Estimation has both theoretical and practical applications. Novel ways to understand harmonic structure are sought after by music theorists, aiding the analysis and composition of pieces. Analysis of music often starts with the manual labeling each chord, which is a time consuming and cogntively demanding expert task. Sequences of chords provide compact representations for use in analysis, music identification and music similarity finding. More broadly speaking, any system that involves the understanding of written music will benefit from chord estimation.

The paper *Modeling and Inferring Proto-voice Structure in Free Polyphony* describes a generative model that encodes the recursive and hierarchical dependencies between notes, giving rise to a grammar-like hierarchical system [9]. This proto-voice model can be used to reduce a piece into a hierarchical structure which encodes an understanding of the tonal/harmonic relations.

Insert descriptive diagram???

Finkensiep suggests in his thesis that the proto-voice model may be an effective way to infer higher level latent entities such as harmonies. Thus, in this project I will answer the question: is this model an effective way to annotate harmonies? By 'effective' we are referring to two things:

- Accuracy: can the model successfully emulate how experts annotate harmonic progressions in musical passages?
- Practicality: can the model be used to do this within a reasonable time frame?

While the original model could in theory be used to generate harmonic annotations, an exhaustive search strategy would be prohibitively time-consuming in practice for any but the shortest musical extracts; one half measure can have over 100,000 valid derivations [8]. My approach will be to explore the use of heuristic search algorithms to solve this problem.

1.2 Related Work

Automatic chord estimation systems first emerged in in the 60's, making use of handcrafted grammar/rule-based systems [26] [46], followed by the development of optimisation algorithms in the early 2000s [31]. In more recent years, supervised learning approaches have have risen in popularity, exploiting large datasets and improved compute power [30] [27] [24].

The proto-voice model is the first to provide a unified theory that relates three aspects of tonal music analysis that are typically considered independently: voice-leading, how notes relate to each other *sequentially*; harmony, how notes relate to each other through *simulataneity*; and note function, how notes relate to each other through recursive *functional dependencies*. Previous models have been developed alongside parsing algorithms to perform automatic chord estimation that consider these dimensions of musical structure separately [26] [46], but in this project we use the relationship between these dimensions of music as the basis of heuristic design.

1.3 Achievements

This was an ambitious project, and I met all of my success criteria and completed the extension tasks. I show that the protovoice model can be used to effectively annotate pieces with chord labels, and these results provide a promising foundation for the model being developed further as a sophisticated tool for the automated analysis of western total music.

Chapter 2

Preparation

In this chapter, I present the work which was undertaken before the code was written. After a brief description of my starting point, I provide an exposition of the Proto-voice Model which forms the foundation of this project. Subsequently, I discuss probabilistic programming and Bayesian inference, including a probabilistic model of harmony. Finally, I describe the software engineering techniques and principles used throughout the project.

2.1 Background Material

2.1.1 Voices

The Protovoice model is concerned with the analysis of Western Classical music, although it could be adapted to different musical styles. For the purposes of this dissertation I will make music theoretical assumptions, but their justification is left to the appendix.

The input we are concerned with is called a score, a symbolic abstraction of a piece of music based on a 2-dimensional axis.

The marks on on score represents notes, with the pitch of the note corresponding to its position on the vertical axis[1], and the notes' position in time represented by the horizontal axis.

The notion of a *voice* is crucial for the understanding of the protovoice model. A voice typically refers to a single melodic line (sequence of notes) that is part of a polyphonic (multiple voices) musical composition. The use of the word is derived from it's use in Bach's four-voice chorales, which consist of 4 sung melodic lines. The term voice is used is used more generally however, the melodic lines do not need to be sung or voice-like in character and can be performed by any melodic instrument.

A piece of music, Explain polyphony and monophony here?

Little diagram of a piece of music, with voices explicitly marked.

Three musical concepts that the protovoice model makes use of:

- **Voice-leading**: What are voices? Implicit vs Explicit voices. Define Stepwise

[1]This is a simplification as there are other factors that determine the pitch, such as the key signature, accidentals and intonation.

- **Harmony**: notes that perceived as sounded simulatiously cause an emergent cognitive phenomonon of harmony, wherein the way the collection of notes sounds together depends on the pitch relationship between notes.

- **Functional Dependencies**: Repetition/ ornamentation.

Insert Diagram demonstrating all three phenoma given a simple example with lines.

2.1.2 The Protovoice Model

The protovoice model is a generative model on the note level of a piece of music, which represents a piece of music as a graph where each note is a node, and notes are connected by stepwise protovoice edges. It generates a piece of music through sequential operations on notes, inserting new notes with edges connected to existing notes.

Inner Operations

A *protovoice* is a sequence of edges between notes based on functional dependencies. The protovoice model is characterised by 3 primitive generative operations on notes.

- Repetitions: a note of the same pitch is repeated before or after a given note

- Neighbor notes: a stepwise ornament to a note.

- Passing notes: notes connecting two protovoices that are separated by a larger interval.

These operations relate notes to one or two *parent* notes, which we can describe as rules. Operations on a single parent are represented by attaching a new *child* note with an edge connected to a parent note:

$$p \implies x \to p \quad \text{or} \quad p \implies p \to x$$

. Operations with two parents are represented by edge replacement.

$$p_1 \to p_2 \quad \implies \quad p_1 \to c \to p_2$$

Outer Operations

To model simulataneity of notes we introduce slices, representing segments of a piece where a group of notes are heard. These provide a higher level abstraction that are used to capture more musical structure, which we refer to as *outer structure*

Diagram showing a slice + a diagram showing a higher level slices, grouping an arpegiation.

A slice m is defined as a multiset of pitches.

A transition $t = (s_l, e, s_r)$ relates two slices with a configuration of edges $e = (e_{reg}, e_{pass})$, a set of regular edges (repetition or neighbor), and a set of passing edges.

Outer operations (Diagram of all three operations):

Split:

$$t \to t'_l s' t'_r$$

Spread:

$$t_l\ s\ t_r \to t'_l\ s'_l\ t'_m\ s'_r\ t'_r$$

Freeze:

$$t \to t$$

Proto-voice harmony

How do we get from a proto-voice (partial)derivation to a harmonic inference?

Explain what harmony is, and how the proto-voice model allows us to capture harmony. Elaboration of the introduction.

What assumptions are needed for a protovoice derivation to be able to describe harmonic entities? These shape heuristic design.

2.1.3 Probabilistic Programming

Provide an explanation of all the concepts I learned and used in this project. Techniques such as marginalisation, joint distributions, bayes rule etc. Probabilistic programming is the combination model definitions and statistical inference algorithms for computing the conditional distribution of inputs (chords) that could have given rise to the observed output (score). We are making the assumption that the score is a realisation of the latent harmonic entities.

Dirchelet distributions Beta distribution Multinomiall distribution Normal Distribution

Inference as model → data → prob distribution → chord guess

2.1.4 Probabilistic Model of Harmony

Outline of the probabilistic model of harmony, describing the parts that are relevant for harmonic annotations. This section allows the reader to understand the evaluation and heuristic modules.

Describe how the parameters were attained

Then describe how to go from the parameters to chord, chordtone and ornamentation distributions

Parameters:

pHarmonies: $\mathbb{N}^{n_c}$
pChordtones: $\mathbb{N}^{n_c}$
pOrnaments: $\mathbb{N}^{n_c}$

Chordtypes, $C = \{$M, m, Mm7, om, o7, mm7, %7, MM7, +, Ger, It, Fr, mM7, +7$\}$

$$\vec{\chi}' \sim \text{Dirchlet}(\text{pHarmonies}, n_c)$$

$$\vec{\chi} = \mathbb{E}(\vec{X}_i) = \frac{\alpha_i}{\sum\limits_j \alpha_j}$$

Chord:

$$c \sim \text{Categorical}(\vec{\chi})$$

Single chordtone distribution. We want to find $P(p|c, ct)$ probability of the pitch given the chord, and that the note is a chordtone:

$$\vec{\phi}'_{ct} \sim \text{Dirchlet}(pChordtones, n_p) \implies \vec{\phi}$$

For each of these parameters we use the MLE to get our probability distribution.

$$\vec{\phi}_{ct} = \text{MLE}(\vec{\phi}'_{ct})$$

$$\vec{\phi}_{or} = \text{MLE}(\vec{\phi}'_{or})$$

$$\vec{\chi} = \text{MLE}(\vec{\chi}')$$

Then for each chord tone,

$$p_{ct} \sim \text{Categorical}(\vec{\phi}_{ct})$$

$$p_{or} \sim \text{Categorical}(\vec{\phi}_{or})$$

We get the distribution of likelihoods for each pitch.

2.1.5 Heuristic Search Algorithms

Provide an outline of the heuristic search paradigm with a formalisation.

Provide a brief overview of different techniques that are used to prune the search space that might be relevant.

2.2 Starting Point

2.2.1 Relevant courses and experience

Haskell I was introduced to Haskell during an internship during the summer before starting this project (July to August 2022). This project is an excuse to learn the language.

Python I have experience coding in Python.

IB Formal Models of Language, Artificial Intelligence.

2.2.2 Existing codebase

NOTE: I need to restructure the codebase to reflect the different modules more clearly. This will help with the flow to the dissertation.

The following describes the protovoices-haskell repository, and where my code contribution will lie:

```
protovoices-haskell
└─ app
   └─ MainExamples.hs
```

```
│
│  │
│  ├─MainISMIR.hs
│  └─...
├─src
│  ├─Heuristics <- My code
│  └─...
├─test
└─...
```

2.3 Requirements Analysis

Table of main components with a dependency and risk analysis

Table 2.1: Overview of main deliverables along with a risk analysis

ID	Delieverable	Priority	Risk
`core1`	Evaluation Module	High	Low
`core2`	End to End Pipeline	High	Medium
`core3`	Parser	High	Medium
`base1`	Random Choice search	High	Low
`base2`	Random Sample	High	Low
`ext1`	Heuristic Search 1	Medium	High
`ext2`	Heuristic Search 2	Medium	High

Short description of where the risk lies.

Pertt chart showing the dependencies between different modules

2.4 Software Engineering Techniques

Justified and documented selection of suitable tools; good engineering approach.

2.4.1 Development model

Usig the dependency and risk analysis above, I created this gantt chart, and totally stuck to it(100% didn't wait until now to get the end-to-end pipeline fully running, and spend most of the time in an extension rabbit hole.. We live and we learn).

Include Gantt chart.

2.4.2 Languages, libraries and tools

The chapter will also cite any new programming languages and systems which had to be learnt

Table 2.2: Languages, libraries and tools

Tool	Purpose	License
Haskell	Main language	...
GHC	Compiling and profiling to inspect time performance and memory usage	GPL-3.0+
Haskell-Musicology	...	...
Dimcat	...	...
Python	...	...
Numpy	...	...
Pandas	...	...
MS3	...	...
Musescore 3	...	...
Protovoice Annotation Tool	...	...
Git	Version Control, Continuous Integration	...

Chapter 3

Implementation

3.1 Repository Overview:

Insert block diagram of components here.

Table 3.1: Repository Overview

File/Folder	Description	LOC
`protovoices-haskell/`	Root directory	2272
`src/`		
`HeuristicParser.hs, HeuristicSearch.hs`	Core Implementation (Section x)	470
`RandomChoiceSearch.hs, RandomSampleParser.hs`	Baseline Implemetation (Section x)	121
`Heuristics.hs, PBHModel.hs`	Extension Implementation (Section x)	383
`FileHandling.hs`	Utilities	188
`...`		
`app/`		
`MainFullParse.hs`	Entry Point	431
`harmonic-inference`		
`experiments/`		
`preprocess.ipynb`		115
`dcml_params.json`		
`inputs/`		
`test/`	Unit Tests (Section x)	611

The following describes an overview of the project repository:

3.2 Core Implementation

3.2.1 Heuristic Parser

This is not a descriptive name. Think of a new name to describe the implementation of the search space of partial reductions. We use the outer representation of structure and outer operations. This is an abstraction.

Parsing Operations

Piece represented by an alternating list of slices and transitions, this is called a path. Define path formally. inductive definitions. dont need the Nothing: just Path trans slice. Transition can be frozen or unfrozen, and boundary or non boundary. Boundary is represented by vertical line, frozen is represented by two lines.

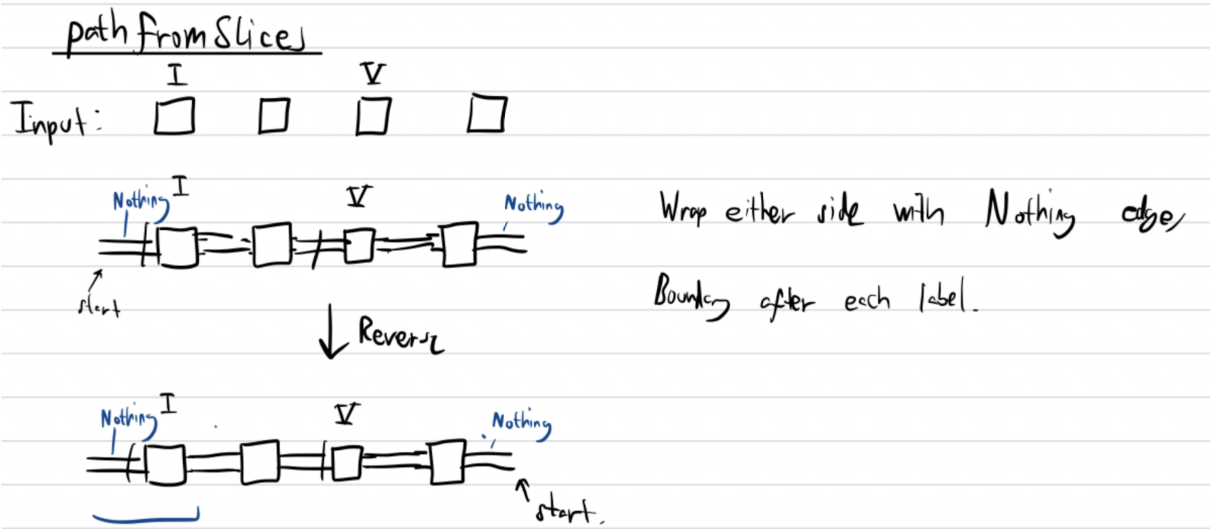

Figure 3.1: Path initiliasation

Definition 3.2.1 (Path)**.** A path is an alternating sequence of an two types of elements, in our case transitions and slices. Definition: Haskell code block or mathematical definition?

Our goal is to reduce the piece into a partial redution by appluying operations until we have one slice per segment. Diagram of this state. This means we have one group of notes per segment, and this group of notes should represent the harmony of the segment.

We parse by applying the inverse of the generative operations, right to left. Unsplit, Unspread, Unfreeze.

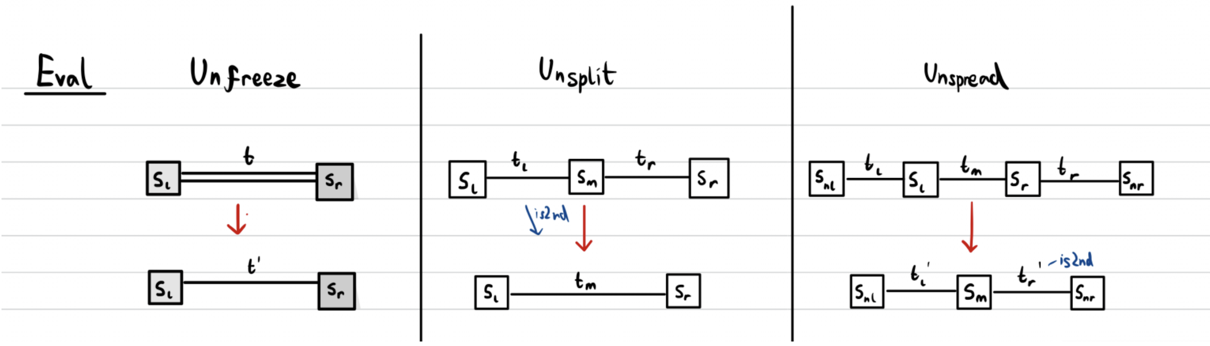

Figure 3.2: Parse operations

State Space

This is how we define the search. We start at the right, the end of the piece. We have a pointer to the current node, and all preceeding slices are open and subsequent slices are frozen. Open Slices can be reduced, but only to the point that there is one slice in a segment. We keep track of the operations performed as it (1). allows us to the draw out the derivation for the partial reduction at the end, and (2). it is used later for calculate a cost for each operation for the heuristic search.

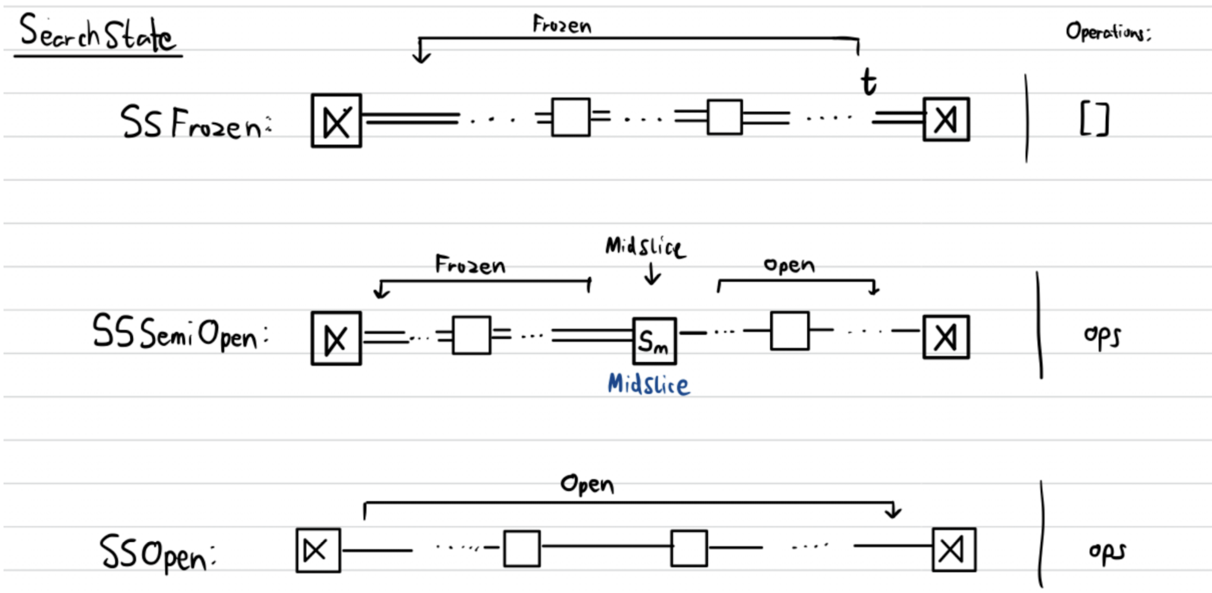

Figure 3.3: Search state

Enumerating State transitions

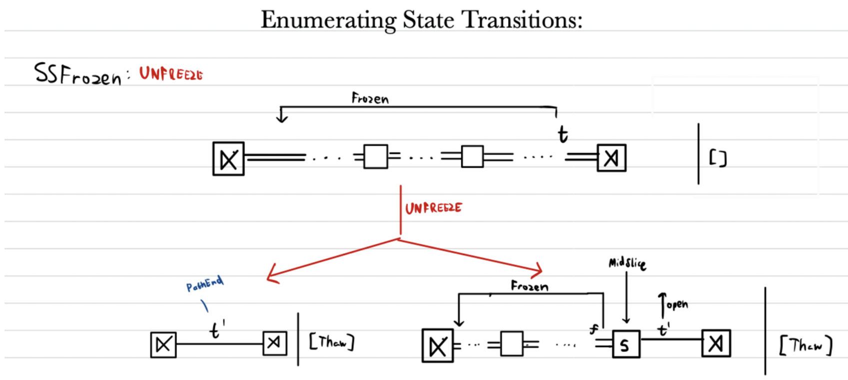

Figure 3.4: Unfreeze operation

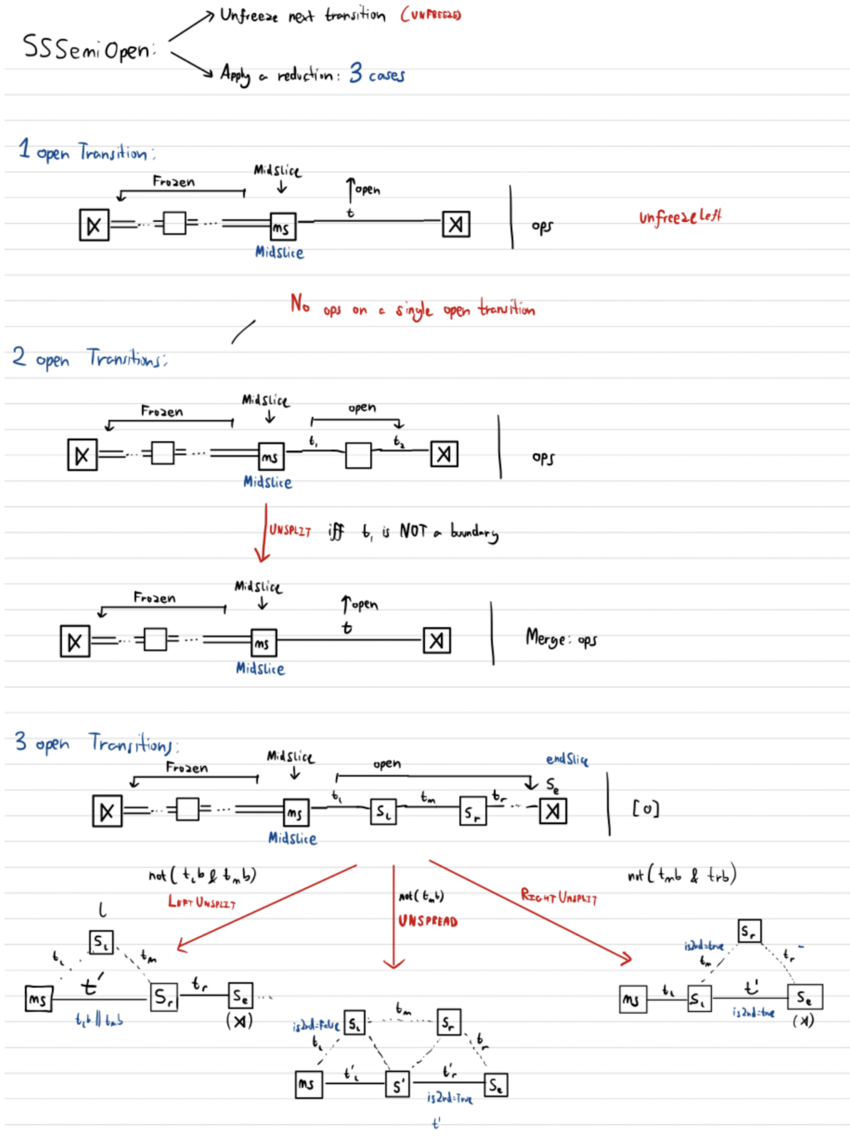

Figure 3.5: Enumeration of operations mid parse. Maybe for appendix? This could be much more concise.

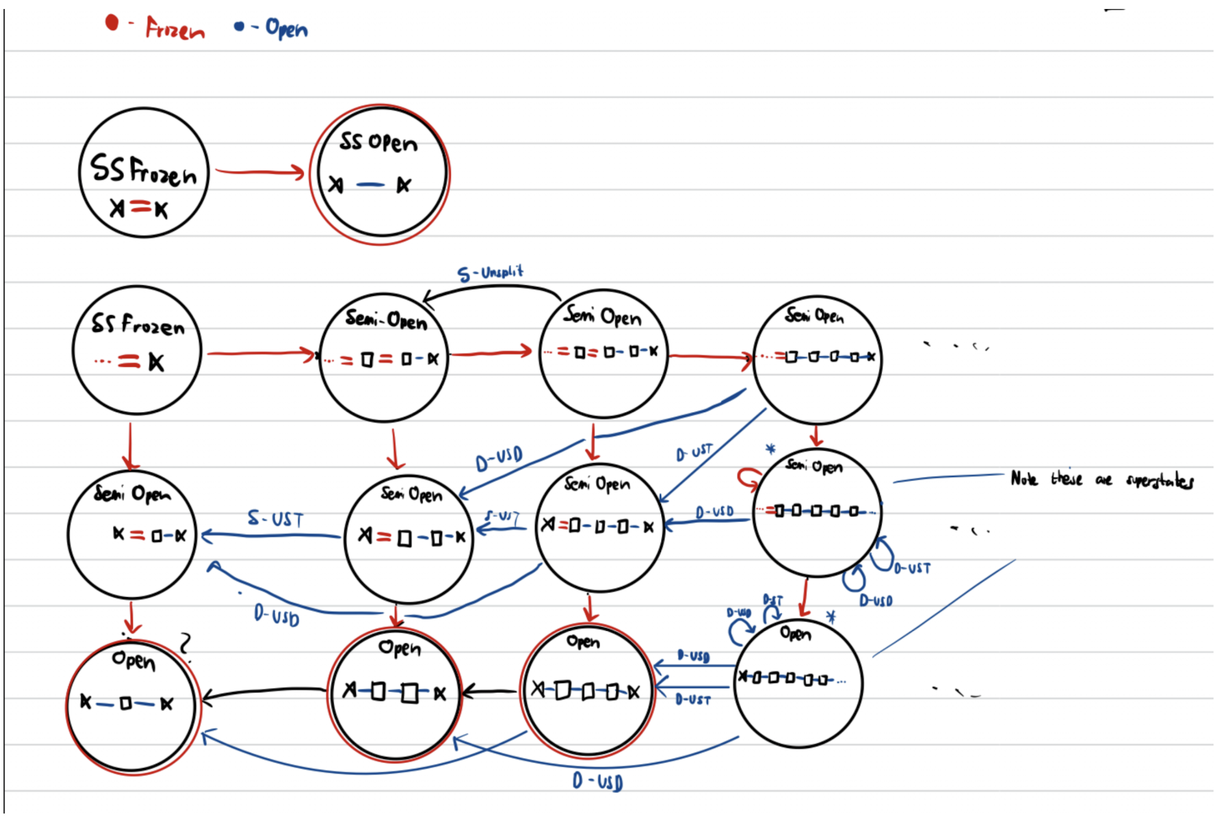

Figure 3.6: State Transition diagram

In the state transition diagram (Figure 3.6), we see all the possible parse states (Is this actually useful? Maybe for appendix). This was useful for me as it helps to conceptualise how the full parse actually works. The dimensions of this digramm of the search state depends on the length of the piece, and the size of each segment. We can see that there is a process of moving to the right to unfreeze transitions, and moving towards the left during reduction operations. Perhaps some simplification of the diagram would be useful. This transition diagram does not consider segment boundaries.

sdfsdfs

sdf

Boundary handling

It is important thhat we don't reduce to an empty segment, because that would mean we've lost all information about the segment, and would not be able to make a harmonic inference. In order to prohibit this, we add additional constraints to the parse operations for each opertation based on the boolean boundary value of all involved transitions.

We use karnaugh maps to determine the boolean expression for these constraints.

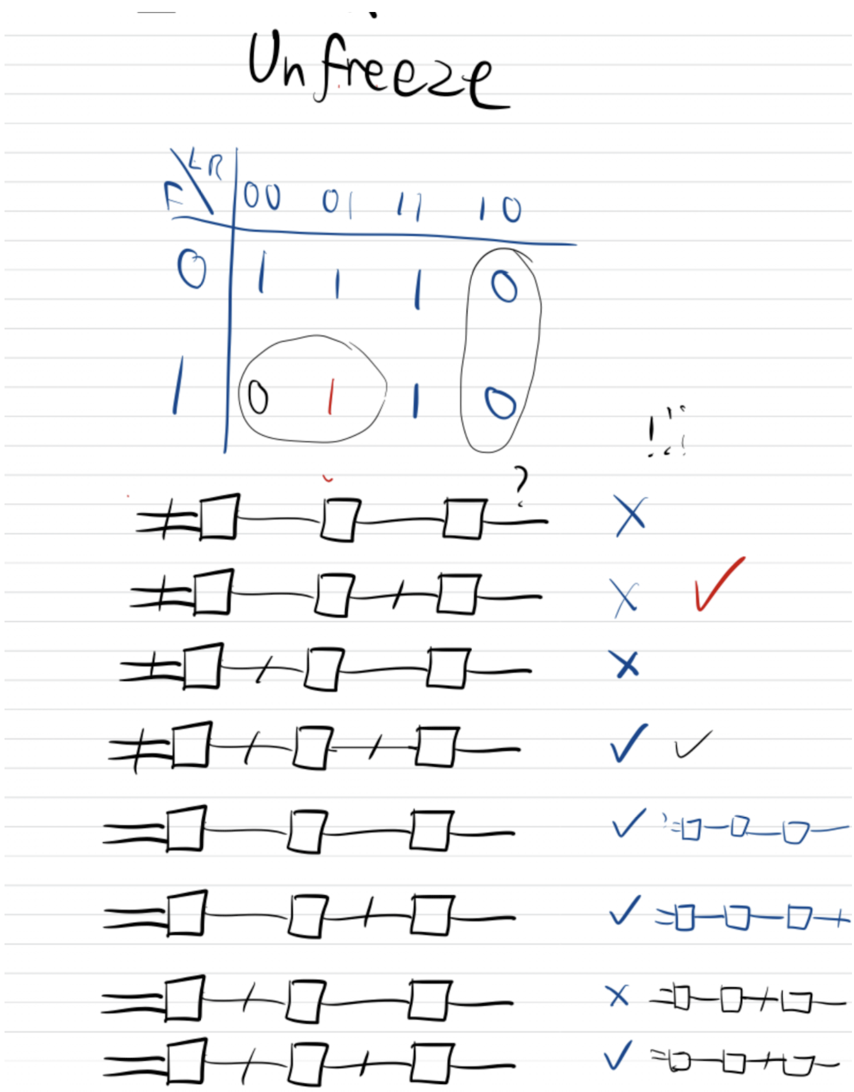

Figure 3.7: Determine boolean boundary expressions for the freeze operations

Could show other maps in the appendix.

3.2.2 Evaluation Module

We need to know exactly what we are trying to achieve before we can understand the baseline and extention implementations.

Probabilistic Model of Harmony

When evaluating using the protovoice model: we assume that we result in only chord tones for each segment. Thus we can use the chord tone probabilities to evaluate the prediction.

When just using a random sample, we have to assume that there is a mixture model of chord tones and ornaments. We can use the learnt parameters to determine the distribution.

These two measures of likelihoods are comparable as they are drawn from the same distributions.

We also need to infer chord labels. We can simply choose the chord that is most likely according to our model.

This gives us two key metrics, likelihood and accuracy.

Could also use a more sophisticated notion of accuracy, using a chord similarity function [15]. The `mir_eval` package provides a plethora of metrics to compare chord label predictions [35].

3.3 Baseline implementation

3.3.1 Random Sample Parser

As a crude baseline we develop two algorithms based on randomly sampling notes for each segment to infer the chord label.

The pure random sample algorithm simply samples random notes for each segment, and uses those to guess the chord label. This doesn't even consider the notes of the piece, so it's really bad, but provides a useful reference.

The per segment sample algorithm samples notes from each segment. Could just sample a random number of notes from each segment, or just use all the notes in the segment to predict the most likely chord label. This is reminiscent of using a key-profile model [42] to find local keys.

3.3.2 Random Choice Search

Now we use our implementation of the protovoice parser, but just do a random walk in the tree of partial reductions. By comparing this against the random ample parser, we can get an idea of the utility of the model. We show that this works surprisingly well.

3.4 Extension Implementation

3.4.1 Heuristic Design

Step 1: Design heuristic to be as accurate as possible. I.e the extreme is to consider every possible parse, but for a single piece there can be over $10^{10^{10^{10^{10^{10}}}}}$ different parses. We consider 1 step at a time at first - this still results in needing to choose an operation out of upwards of 30,000,000 options for just a single step.

First the full piece heuristic parse

Problem of very large slices.

Segment by segment heuristic parse - avoids the problem, but is slightly hacky. Can we incorprate our knowledge regarding the relative proportion of chord tones and ornaments. Should we allow duplicates of notes in slices? Perhaps we should favour spreads more.

Always consider a certain number of slices and spreads.

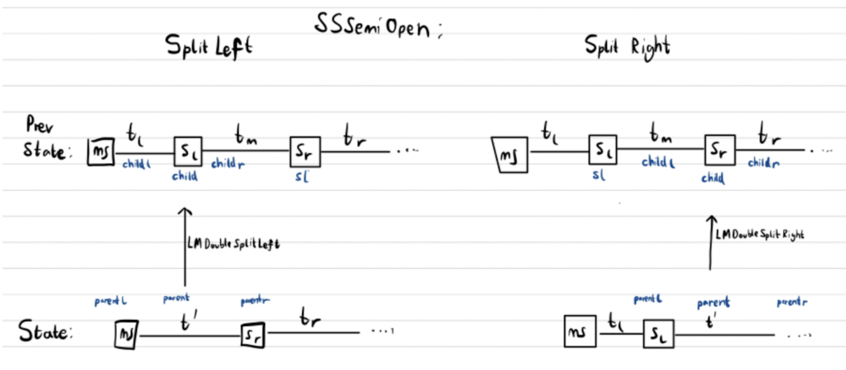

Figure 3.8: Split operation

Scoring Unsplit Operations

Consider the Split rule :

$$t \to t_l' \; s' \; t_r'$$

During a split, each edge in the transition and each node in an adjacent slice can be elaborated by one or more inner operations. These new edges can be discarded or kept to form the new edge of t_l' and t_r'.

The notes in the child slice s can either have edges connected to the left neighboring slice or right neighbouring slice, or both. I.e for each note in the child slice, it can be a an ornmentation of a previous note, subsequent note, both, or repetition of prev note, subsequent note etc. So we consider the chord tone profiles of the involved slices.

We first guess the chord type each parent slice.

$$\theta_l = \underset{c \in C}{argmax}\, P(s_l|c) \quad , \quad \theta_r = \underset{c \in C}{argmax}\, P(s_r|c)'$$

We now consider each edge individually, considering their likelihoods based on the proabilistic model of harmony along with theoretical assumptions.

Single Sided Operations

- Right Neighbour (Left Neighbour anagolously)

$$x \implies x \to n \quad , x, n \in P$$

$$x \sim \text{Categorical}(\sigma_{ct}^{\theta_l})$$

$$n \sim \text{Categorical}(\sigma_{or}^{\theta_r})$$

  Find

$$P(x, n \mid \theta_l)$$

- Right Repeat (Left Repeat anagolously)

$$x \implies x \to x \quad , x \in P$$

$$x \sim \text{Categorical}(\sigma_{ct}^{\theta_l})$$

  Find

$$P(x \mid \theta_l)$$

Two Sided Operations

- Root Note: This operation is only done once in the original model. In our case we do not need to consider due to segment boundaries.

- Full Repeat:

$$x \implies x \to n \quad , x, n \in P$$

$$x \sim \text{Categorical}(\sigma_{ct}^{\theta_l})$$

$$n \sim \text{Categorical}(\sigma_{or}^{\theta_r})$$

  Find

$$P(x, n \mid \theta_l)$$

- Left Repeat of Right:

$$x \to y \implies x \to y' \to y$$

$$y \sim \text{Categorical}(\sigma_{ct}^{\theta_l})$$

  Find

$$P(y \mid \theta_l)$$

- Full Neighbour:

$$x_1 \to x_2 \implies x_1 \to n \to x_2, x \in P$$

  Find

$$P(\mid \theta_l, \theta_r)$$

Scoring Unspread Operations

Consider the Spread rule :

$$t_l s_r \to t'_l s_l t'_m s_r t'_r$$

We make the assumption that s, s_l, & s_r are all realisations of the same chord. This lines up with the music theorretical basis for this operation in the model(justify).

Thus we find the most likely chord (optional extension: marginalise over all chords)

$$\theta = \underset{c \in C}{argmax}\, P(s|c)$$

When then measure the extent to which the parent slics match this chord.

$$p(s_l, s_r|\theta)$$

We can calculate $p(s_l|\theta)$ and $p(s_r|\theta)$ using the multinomial distribution probability density function as described in the preparation chapter.

Scoring Unfreeze Operations

We assign 0 cost to unfreeze operations. This means we need to be careful about ensure that we don't just unfreeze the entire piece immediately. Careful construction of the search algorithm can ensure this. More later.

Full state evalutation

We need to combine all of these in a fair way. Also the distinction between splits and spreads need to be considred, as they are different operations, the calculations of likelihood may cause an imbalance. All likelihoods are stored in log space.

3.4.2 Heuristic Search

Step 2: Relax the heuristic search in order to reduce runtime/ lower complexity.

In the case that there are 85,000,000 options, perhaps we should sample the options rather than evaluating all of them.

This version of heuristic search should be able to parse full pieces (hopefully), so can be used to compare with the baselines on an entire corpus.

Beam of size n, with 1 for a freeze, k for spread, n-k-1

3.5 Testing

Show unit tests, and examples of the test/development cycle for the heuristic search development

Chapter 4

Evaluation

In this chapter, I provide qualitative and quantitative evaluations of the work completed. I then provide and interpret evidence to show that the success criteria were met.

The main questions to answer are as follows:

- *Can the proto-voice model be used to accurately infer chord labels?*
- *Can the proto-voice model be used to practically infer chord labels?*
- *How well my heuristic search algorithms infer chord labels?*

4.1 Accuracy

Things to note

- The fact that segmentation is known ahead of time provides a great deal of information [12]
- So we can use comparisons between the random sample from each segment algorithm and the random parse algorithm to see if the use of the grammar provides an advantage over just sampling the notes directly, without looking at relations between notes.
- Then we want a heuristic search algorithm that considers each option exhaustively and finds the best local option. This is too computationally expensive to be used for whole pieces.
- Given there can be millions of possible next states in the search, we need to look at different strategies to avoid searching through them all. E.g just sample states.
- Sensitivity Analysis for the heuristic search is useful for the evaluation. Explore how robust it is to handcrafted attacks/ different types of passages.
- Could evaluate by segments instead of pieces.

4.2 Performance

4.3 Heuristic Search (Extension)

4.4 Success Criteria

4.5 Limitations

Chapter 5

Conclusions

In this chapter, I first discuss the success achieved by the project then offer a reflection on lessons learned. Finally, I consider the directions in which there is potential for future work.

5.1 Achievements

5.2 Lessons learned

5.3 Future Work

[?]

Bibliography

[1] Chen, T.-P., and Su, L. Functional Harmony Recognition of Symbolic Music Data with Multi-task Recurrent Neural Networks, Sept. 2018.

[2] Chen, T.-P., and Su, L. Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition. In *International Society for Music Information Retrieval Conference* (2019).

[3] Cohen, D. E. "The Imperfect Seeks Its Perfection": Harmonic Progression, Directed Motion, and Aristotelian Physics. *Music Theory Spectrum 23*, 2 (Oct. 2001), 139–169.

[4] Davidson-Pilon, C. Bayesian Methods for Hackers, Mar. 2023.

[5] Deng, J., and Kwok, Y.-K. Large vocabulary automatic chord estimation using bidirectional long short-term memory recurrent neural network with even chance training. *Journal of New Music Research 47*, 1 (Jan. 2018), 53–67.

[6] Ebcioğlu, K. An Expert System for Harmonizing Four-Part Chorales. *Computer Music Journal 12*, 3 (1988), 43–51.

[7] Feisthauer, L., Bigo, L., Giraud, M., and Levé, F. Estimating keys and modulations in musical pieces. In *Sound and Music Computing Conference (SMC 2020)* (June 2020).

[8] Finkensiep, C. *The Structure of Free Polyphony.* PhD thesis, EPFL, Lausanne, 2023.

[9] Finkensiep, C., and Rohrmeier, M. Modeling and Inferring Proto-voice Structure in Free Polyphony. In *Proceedings of the 22nd ISMIR Conference* (Online, Nov. 2021).

[10] Gjerdingen, R. Cognitive Foundations of Musical Pitch Carol L. Krumhansl. *Music Perception: An Interdisciplinary Journal 9* (July 1992), 476–492.

[11] Goodman, J. Semiring Parsing. *Computational Linguistics 25*, 4 (1999), 573–606.

[12] Gotham, M., Kleinertz, R., Weiss, C., Müller, M., and Klauk, S. What if the 'When' Implies the 'What'?: Human harmonic analysis datasets clarify the relative role of the separate steps in automatic tonal analysis. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*

2021, Online, November 7-12, 2021 (2021), J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., pp. 229–236.

[13] Granroth-Wilding, M. Harmonic Analysis of Music Using Combinatory Categorial Grammar. *The University Of Edinburgh* (2013).

[14] Harasim, D. Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony. In *Proceedings of the 20th ISMIR Conference* (2019), T. J. O'Donnell and M. A. Rohrmeier, Eds., ISMIR.

[15] Humphrey, E. J., and Bello, J. P. Four Timely Insights on Automatic Chord Estimation.

[16] Johannes, H. Ms3 - Parsing MuseScore 3. https://github.com/johentsch/ms3, 2021.

[17] Kidney, D. O., and Wu, N. Algebras for weighted search. *Proceedings of the ACM on Programming Languages 5*, ICFP (Aug. 2021), 1–30.

[18] Koops, H. V., de Haas, W. B., Bransen, J., and Volk, A. Automatic chord label personalization through deep learning of shared harmonic interval profiles. *Neural Computing and Applications 32*, 4 (Feb. 2020), 929–939.

[19] Koops, H. V., de Haas, W. B., Burgoyne, J. A., Bransen, J., Kent-Muller, A., and Volk, A. Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research 48*, 3 (May 2019), 232–252.

[20] Korzeniowski, F., and Widmer, G. Improved Chord Recognition by Combining Duration and Harmonic Language Models, Aug. 2018.

[21] Krumhansl, C. L., and Kessler, E. J. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review 89* (1982), 334–368.

[22] Lerdahl, F., and Jackendoff, R. *A Generative Theory of Tonal Music*, repr. ed. MIT Press, Cambridge, Mass., 2010.

[23] Marsden, A. Schenkerian Analysis by Computer: A Proof of Concept. *Journal of New Music Research 39*, 3 (Sept. 2010), 269–289.

[24] Masada, K., and Bunescu, R. Chord Recognition in Symbolic Music: A Segmental CRF Model, Segment-Level Features, and Comparative Evaluations on Classical and Popular Music, Oct. 2018.

[25] Mauch, M., Mullensiefen, D., Dixon, S., and Wiggins, G. Can Statistical Language Models be used for the Analysis of Harmonic Progressions?

[26] Maxwell, H. J. An expert system for harmonizing analysis of tonal music. In *Understanding Music with AI: Perspectives on Music Cognition.* MIT Press, Cambridge, MA, USA, Aug. 1992, pp. 334–353.

[27] McLeod, A., and Rohrmeier, M. A Modular System for the Harmonic Analysis of Musical Scores using a Large Vocabulary. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021* (2021), J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., pp. 435–442.

[28] Mearns, L. The Computational Analysis of Harmony in Western Art Music.

[29] Neuwirth, M., Harasim, D., Moss, F. C., and Rohrmeier, M. The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets. *Frontiers in Digital Humanities 5* (July 2018), 16.

[30] Ni, Y., Mcvicar, M., Santos-Rodriguez, R., and De Bie, T. An end-to-end machine learning system for harmonic analysis of music, July 2011.

[31] Pardo, B., and Birmingham, W. P. Algorithms for Chordal Analysis. *Computer Music Journal 26*, 2 (June 2002), 27–49.

[32] Pickens, J. *Harmony Modeling for Polyphonic Music Retrieval.* PhD thesis, University of Massachusetts, 2004.

[33] Quinn, I. Are Pitch-Class Profiles Really "Key for Key"? *Zeitschrift der Gesellschaft für Musiktheorie [Journal of the German-Speaking Society of Music Theory] 7* (Jan. 2010).

[34] Radicioni, D. P., and Esposito, R. BREVE: An HMPerceptron-Based Chord Recognition System. In *Advances in Music Information Retrieval*, J. Kacprzyk, Z. W. Raś, and A. A. Wieczorkowska, Eds., vol. 274. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 143–164.

[35] Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., and Ellis, D. P. W. Mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proceedings of the 15th International Conference on Music Information Retrieval* (2014).

[36] Raphael, C., and Stoddard, J. Functional Harmonic Analysis Using Probabilistic Models. *Computer Music Journal 28*, 3 (Sept. 2004), 45–52.

[37] Rocher, T., Robine, M., Hanna, P., and Strandh, R. Dynamic Chord Analysis for Symbolic Music.

[38] Sapp, C. Visual hierarchical key analysis. *Computers in Entertainment 3* (Oct. 2005), 1–19.

[39] Sapp, C. 6 Computational Chord-Root Identification in Symbolic Musical Data: Rationale, Methods, and Applications.

[40] Sidorov, K., Jones, A., and Marshall, D. MUSIC ANALYSIS AS A SMALLEST GRAMMAR PROBLEM.

[41] Temperley, D. An Algorithm for Harmonic Analysis. *Music Perception 15*, 1 (Oct. 1997), 31–68.

[42] Temperley, D. A Bayesian Approach to Key-Finding. In *Music and Artificial Intelligence*, G. Goos, J. Hartmanis, J. van Leeuwen, C. Anagnostopoulou, M. Ferrand, and A. Smaill, Eds., vol. 2445. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 195–206.

[43] Temperley, D. A Bayesian Approach to Key-Finding. In *Music and Artificial Intelligence* (Berlin, Heidelberg, 2002), C. Anagnostopoulou, M. Ferrand, and A. Smaill, Eds., Lecture Notes in Computer Science, Springer, pp. 195–206.

[44] Temperley, D. A Unified Probabilistic Model for Polyphonic Music Analysis. *Journal of New Music Research 38*, 1 (Mar. 2009), 3–18.

[45] Volk, A. Improving Audio Chord Estimation by Alignment and Integration of Crowd-Sourced Symbolic Music. 141–155.

[46] Winograd, T. Linguistics and the Computer Analysis of Tonal Harmony. *Journal of Music Theory 12*, 1 (1968), 2–49.

Appendix A

Additional Information

Appendix B

Project Proposal

Inferring Harmony from Free Polyphony

Judah Daniels

April 5, 2023

DOS: Prof. Larry Paulson

Crsid: jasd6
College: Clare College

B.1 Abstract

A piece of music can be described using a sequence of chords, representing a higher level harmonic structure of a piece. There is a small, finite set of chord types, but each chord can be realised on the musical surface in a practically infinite number of ways. Given a score, we wish to infer the underlying chord types.

The paper *Modeling and Inferring Proto-voice Structure in Free Polyphony* describes a generative model that encodes the recursive and hierarchical dependencies between notes, giving rise to a grammar-like hierarchical system [?]. This proto-voice model can be used to reduce a piece into a hierarchical structure which encodes an understanding of the tonal/harmonic relations of a piece.

Christoph Finkensiep suggests in his paper that the proto-voice model may be an effective way to infer higher level latent entities, such as harmonies or voice leading schemata. Thus in this project I will ask the question: is this parsing model an effective way to annotate harmonies? By ‘effective’ we are referring to two things:

- Accuracy: can the model successfully emulate how experts annotate harmonic progressions in musical passages?
- Practicality: can the model be used to do this within a reasonable time frame?

While the original model could in theory be used to generate harmonic annotations, its exhaustive search strategy would be prohibitively time-consuming in practice for any but the shortest musical extracts; one half measure can have over 100,000 valid derivations [?]. My approach will be to explore the use of heuristic search algorithms to solve this problem.

B.2 Substance and Structure

B.2.1 Core: Search

The core of this project is essentially a search problem characterised as follows:

- The state space S is the set of all possible partial reductions of a piece along with each reduction step that has been done so far.
- We have an initial state $s_o \in S$, which is the empty reduction, corresponding to the unreduced surface of the piece. The score is represented as a sequence of slices grouping notes that sound simultaneously. We are also given the segmentation of the original chord labels that we wish to retrieve.
- We have a set of actions, A modelled by a function $action : A \times S \to S$. These actions correspond to a single reduction step.
  - The reduction steps are the inverses of the operations defined by the generative proto-voice model.
- Finally we have a goal test, $goal : S \to \{true, false\}$ which is true iff the partial reduction s has exactly one slice per segment of the input.

  - This means the partial reduction s contains a sequence of slices which start and end positions corresponding to the segmentation of the piece.

- At the first stage, this will be implemented using a random graph search algorithm, picking each action randomly, according to precomputed distributions.

B.2.2 Core: Evaluation

The second core task is to create an evaluation module that iterates over the test dataset, and evaluates the partial reduction computed by the search algorithm above. This will be done by comparing the outputs to ground truth annotations from the Annotated Beethoven Corpus.

In order to do this I will make use of the statistical harmony model from Finkensiep's thesis, *The Structure of Free Polyphony* [**?**]. This model provides a way of mapping between the slices that the algorithm generates and the chords in the ground truth. This can be used to empirically measure how closely the slices match the expert annotations.

B.2.3 Extension

Once the base search implementation and evaluation module have been completed, the search problem will be tackled by heuristic search methods, with different heuristics to be trialled and evaluated against each other. The heuristics will make use of the chord profiles from Finkensiep's statistical harmony model discussed above. These profiles relate note choices to the underlying harmony. Hence the heuristics may include:

- How the chord types relate to the pitches used.

- How the chord types relate which notes are used as ornamentation, and the degree of ornamentation.

- Contextual information about neighboring slices

B.2.4 Overview

The main work packages are as follows:

Preliminary Reading – Familiarise myself with the proto-voice model, and read up on similar models and their implementations. Study heuristic search algorithms.

Dataset Preparation – Pre-process the Annotated Beethoven Corpus into a suitable representation for my algorithm.

Basic Search – Implement a basic random search algorithm that takes in surface and segmentations, and outputting the sequence of slices matching the segmentations.

Evaluation Module – Implement an evaluation module to evaluate the output from the search algorithm.

End-to-end pipeline – Implement a full pipeline from the data to the evaluation that can be used to compare different reductions.

Heuristic Design – Extension – Trial different heuristics and evaluate their performance against each other.

Dissertation – I intend to work on the dissertation throughout the duration of the project. I will then focus on completing and polishing the project upon completion.

B.3 Starting Point

The following describes existing code and languages that will be used for this project:

Haskell – I will be using Haskell for this project as it is used in the proto-voice implementation. It must be noted that my experience with Haskell is limited, as I was first introduced to it via an internship this summer (July to August 2022).

Python – Python will be used for data handling. I have experience coding in Python.

Prior Research - Over the summer I have been reading the literature on computational models of music, as well as various parsing algorithms such as semi-ring parsing [?], and the CYK algorithm, which is used in the implementation of the proto-voice model.

Protovoices-Haskell – The paper *Modeling and Inferring Proto-Voice Structure in Free Polyphony* [?] includes an implementation of the proto-voice model in Haskell. A fork of this repository will form the basis of my project. This repository includes as parsing module which will be used to perform the actions in the search space of partial reductions. There is module that can exhaustively enumerate reductions of a piece, but this is infeasible in practice due to the blowup of the derivation forest.

MS3 – This is a library for parsing MuseScore Files and manipulating labels [?], which I will use as part of the data processing pipeline.

ABC – The *Annotated Beethoven Corpus* [?] contains analyses of all Beethoven string quartets composed between 1800 and 1826), encoded in a human and machine readable format. This will be used as a dataset for this project.

B.4 Success Criteria

This project will be deemed a success if I complete the following tasks:

- Develop a baseline search algorithm that uses the proto-voice model to output a partial reduction of a piece of music up to the chord labels.
- Create an evaluation module that can take the output of the search algorithm and quantitatively evaluate its accuracy against the ground truth annotations by providing a score based on a statistical harmony model.
- Extension: Develop one or more search algorithms that use additional heuristics to inform the search, and compare the accuracy with the baseline algorithm.

B.5 Timetable

Time frame	Work	Evidence
Michaelmas (Oct 4 to Dec 2)		
Oct 14 to Oct 24	*Oct 14*: Final proposal deadline. Preparation work: familiarise myself with the dataset and the proto-voice model implementation. Work on manipulating reductions using the proto-voice parser provided by the paper.	None
Oct 24 to Nov 7	Dataset preparation and handling.	Plot useful metrics about the dataset using Haskell
Nov 7 to Nov 21	Random Search implementation	None
Nov 21 to Dec 5	Evaluation Module. Continue with search implementation.	Evaluate a manually created derivation and plot results
Vacation (Dec 3 to Jan 16)		
Dec 5 to Dec 11	Evaluate performance of random search. Begin to work on extensions	Plot results
Dec 10 to Dec 21	Trial different heuristics. Implement an end-to-end pipeline from input to evaluation.	None
Dec 21 to Dec 27	None	None
Dec 27 to Jan 10	Continue trialing and evaluating heuristics	*Fulfill success criterion: At least one heuristic technique gives better performance than random search.*
Lent (Jan 17 to Mar 17)		
Jan 4 to Jan 20	Buffer Period to help keep on track	None
Jan 20 to Feb 3	*Feb 3*: Progress Report Deadline. Write progress report and prepare presentation. Write draft *Evaluation* chapter	Progress Report (approx. 1 page)
Feb 3 to Feb 17	Prepare presentation.	*Feb 8 – 15*: Progress Report presentation
Feb 17 to Mar 3	*Feb 17*: How to write a Dissertation briefing. Write draft Introduction and Preparation chapters. Incorporate feedback on Evaluation chapter.	Send draft Introduction and Preparation chapter to supervisor

B.6 Resources

I plan to use my own laptop for development: MacBook Pro 16-inch, M1 Max, 32GB Ram, 1TB SSD, 24-core GPU.

All code will be stored on a GitHub repository, which will guarantee protection from data loss. I will easily be able to switch to using university provided computers upon hardware/software failure.

The project will be built upon work that has been done in the DCML (Digital cognitive musicology lab) based in EPFL. The files are in their Github repository, and I have been granted permission to access their in-house datasets of score annotations, as well as software packages which are used to handle the data.

B.7 Supervisor Information

Peter Harrison, head of Centre for Music and Science at Cambridge, has agreed to supervise me for this. We have agreed on a timetable for supervisions for this year. I am also working with Christoph Finkensiep, a PHD student at the DCML, and originator of the proto-voice model. Professor Larry Paulson has agreed to be the representative university teaching officer.