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 10, 2023

Proforma

Candidate Number:	**2200D**
Project Title:	**Inferring Harmony from Free Polyphony**
Examination:	**Computer Science Tripos – Part II, July, 2023**
Word Count:	**5434**[1]
Code Line Count:	**2272**[2]
Project Originator:	**Christoph Finkensiep**
Supervisor:	**Dr Peter Harrison**

Original Aims of the Project

Work Completed

Special Difficulties

There were no special difficulties encountered in this project

[1]This word count was computed by `texcount main.tex`
[2]This code line count was computed by using `cloc`

Contents

List of Figures

Acknowledgements

I'd like to thank Christoph Finkensiep and Peter Harrison for being awesome.

Introduction

1.1 Motivation and Aim

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 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 S (a symbolic representation of a piece of music), we wish to infer the sequence of underlying chord types $L = l_0, \ldots, l_n$.

$$\hat{L} = \arg\max_{L} p(S|L) \tag{1.1}$$

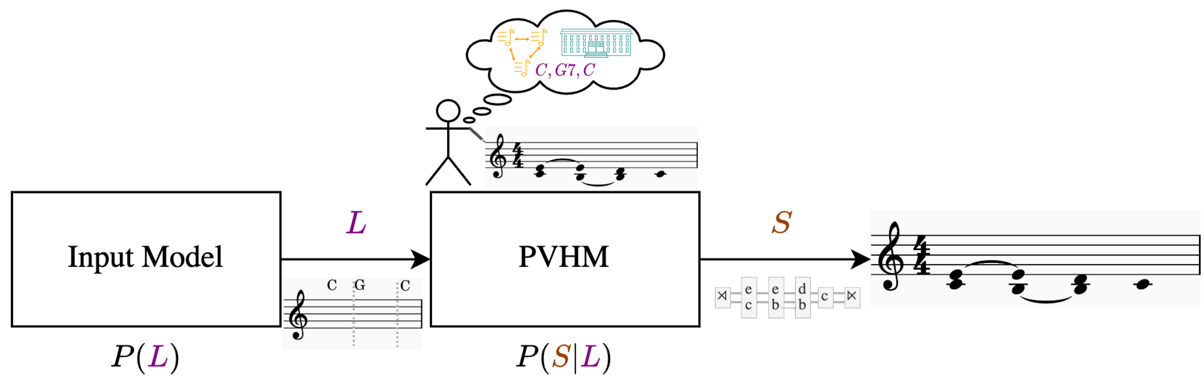
We present a **novel solution**, the **ProtoVoice Harmony Model (PVHM)** that infers the sequence of chord types given score using probabilistic models of latent structure and harmony, dimensionality reduction, parsing strategies and heuristic search methods.

Automatic Chord Estimation has both theoretical and practical applications. 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 tonal music will benefit from chord estimation.

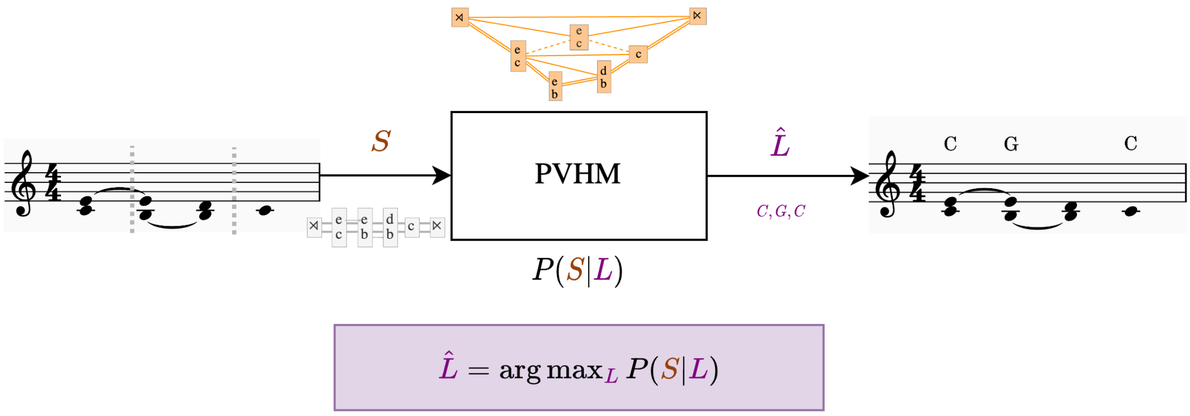
The paper *Modeling and Inferring Protovoice Structure in Free Polyphony* describes a generative model that encodes the recursive and hierarchical dependencies between notes, giving rise to a grammar-like hierarchical system [10]. This model can be used to reduce a piece into a hierarchical structure which encodes an understanding of the tonal/harmonic relations.

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].

The **PVHM** uses *heuristic search* strategies to infer *latent structure* using the protovoice model, followed by *feature extraction* which finally allows the chord labels to be inferred using a *probabilistic model of harmony.*



(a) Modelling



(b) Inference

Figure 1.1: Problem overview

1.2 Related Work

Automatic chord estimation systems first emerged in in the 60's, making use of hand-crafted grammar/rule-based systems [27] [48], followed by the development of optimisation algorithms in the early 2000s [32]. In more recent years, supervised learning approaches have have risen in popularity, exploiting large datasets and improved compute power [31] [28] [25].

The protovoice 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 [27] [48], 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; I met all the 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. The PVHM has been made **open source** to accommodate future research in the area.

Preparation

2.1 Bayesian Inference

2.1.1 Bayesian Inference and Modelling

Bayesian inference is a statistical inference paradigm in which Bayes' theorem is used to update the probability for a hypothesis in light of available evidence. This allows us to *infer* information about *latent*(hidden) variables based on *observed* evidence.

The method of Bayesian inference reduces to representing the *joint distribution* of all the random variables in a system, such that we can compute any probability of interest in that system. Let X and Z denote the observed and latent(unobserved) variables respectively, and let Y denote the random variable we wish to infer. Then we are interested in representing the *joint distribution* $P(X, Y, Z)$. **Modelling** is the process of approximating a joint distribution through abstract representations and relationships.

We first consider how we would compute the best hypothesis for Y given the evidence X, ignoring the latent variables Z.

There are two methods used to find the most likely hypothesis Y:

- The **maximum a priori estimate** maximises the *conditional likelihood* of the hypothesis given the evidence, given by $P(Y|X)$. This requires us to know the *prior* $P(Y)$, the distribution of the hypothesis, beforehand. The prior can be learned from labeled data.

$$\begin{aligned}\hat{Y} &= \arg\max_Y P(Y|X) \\ &= \arg\max_Y P(X|Y)P(Y)\end{aligned} \tag{2.1}$$

- The **maximum likelihood estimate** maximises the conditional likelihood of the observed evidence given the hypothesis, given by $P(X|Y)$. This method allows us to capture *uncertainty* in the prediction, given by $P(X|\hat{Y})$.

$$\hat{Y} = \arg\max_Y P(X|Y) \tag{2.2}$$

Definition 2.1.1 (Factoring)**.**

$$P(A, B) = P(B|A)\ P(A) \tag{2.3}$$

Definition 2.1.2 (Marginalisation)**.**

$$P(A) = \sum_{B} P(A|B)\ P(B) \tag{2.4}$$

Definition 2.1.3 (Chain Rule of Probability)**.**

$$P(A, B) = P(A|B)\ P(B) \tag{2.5}$$

2.1.2 Inferring Latent variables

To incorporate the latent variables Z, we use *marginalisation* and the *chain rule of probability* to show that:

$$\begin{aligned} P(X|Y) &= \sum_{Z} P(X, Z|Y) \\ &= \sum_{Z} P(X|Y, Z)\ P(Z|Y) \end{aligned} \tag{2.6}$$

Given a model of $P(Z|Y)$, we avoid summing over the values of Z by inferring $\hat{Z}$, the most likely value of Z at $\hat{Y}$. The value $\hat{Z}$ is inferred using MLE or MAP estimation based on a model of $P(Z|Y)$.

We update the prior $P(Z|Y)$ as:

$$P(Z|Y) = \begin{cases} 1 & \text{if } Z = \hat{Z} \\ 0 & \text{otherwise} \end{cases}$$

This gives us an approximation of $P(X|Y)$:

$$P(X|Y) \approx P(X|Y, \hat{Z}) \tag{2.7}$$

2.2 Overview of Approach

Probabilistic programming is a programming paradigm that makes use of model definitions and statistical inference algorithms to compute the conditional distribution of inputs that could have given rise to an observed output.

In the context of ACE, we consider the underlying sequence of chord labels $L_0, L_1, \ldots L_n$ as an **input**, and the musical surface or score as the **observed output** S.

In this sense, ACE can be solved by finding the most likely sequence of labels for the given surface, described by the equation:

$$\hat{L} = \arg\max_{L} P\left(S|L\right) \tag{2.8}$$

The difficulty arises from the complexity and prohibitively large number of the **latent variables** ϕ; in reality, we need to maximise $\sum_{\phi} P(S|L, \phi)\ P(\phi|L)$.

$$\hat{L} = \arg\max_{L} \sum_{\phi} P(S|L, \phi)\ P(\phi|L) \tag{2.9}$$

The set of latent variables ϕ is **practically infinite**. These include the author's compositional conception, their musical conception, cognitive phenomena experienced by listeners, shared experience distilled into music theory, musical trends/culture and notational conventions.

This cannot be solved analytically, but we approximate $\hat{L}$ using models that encode domain specific knowledge about both the music generation and labelling processes, through joint conditional distributions.

Approximating Conditional Likelihood with MLE

Now consider the set of latent RVs ϕ as the union of two *disjoint sets*, ϕ' and ϕ'', where ϕ' is the subset of ϕ which we can *reasonably infer* given L and S, and $\phi'' = \phi \setminus \phi'$. Assuming $P(\phi'|L)$ and $P(\phi''|L)$ are independent, it follows:

$$\begin{aligned} P(S|L) &= \sum_{\phi'} \sum_{\phi''} P(S|L, \phi', \phi'')\ P(\phi', \phi''|L) \\ &= \sum_{\phi'} \sum_{\phi''} P(S|L, \phi', \phi'')\ P(\phi'|L)\ P(\phi''|L) \end{aligned} \tag{2.10}$$

We take $P(\phi''|L)$ to be a uniform distribution as we have no prior knowledge of the latent variables that we cannot infer. Using the technique described in Equation 2.7, we find $\hat{\phi}'$, thus giving:

$$\hat{L} \approx \arg\max_{L} P(S|L, \hat{\phi}') \tag{2.11}$$

2.3 Inferring Structure

2.3.1 The Protovoice Model

The protovoice model is a formal generative model which represents a piece of music as a graph where each note is a node, and notes are connected by stepwise protovoice edges.

The Protovoice model is primarily concerned with the analysis of Western Classical music, although it could be applied to different musical styles, such as jazz or some popular western music [8].

2.3.2 Voices

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.

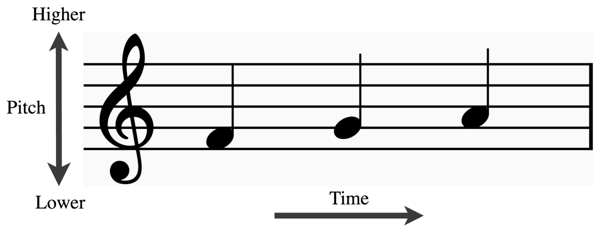


Figure 2.1: An example of music notation showing an ascending stepwise sequence.

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 musical composition. The term is derived from its use in choral music, such as J.S 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.

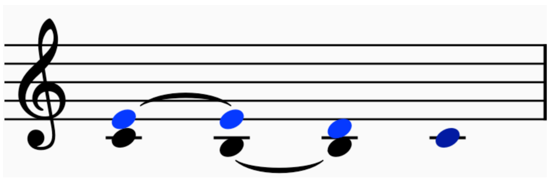


Figure 2.2: A short cadential phrase with two voices.

Polyphony refers to a piece of music that can contains more than one voice. Typically polyphonic music will have a set number of voices throughout the piece, but *free polyphony* refers to music where the number of voices is arbitrary and can change throughout the piece.

There are three types of relations between notes that form the basis for the protovoice model:

- **Horizontal**: As music is perceived in time, natural sequential relations arise between subsequent notes, in fact, we can define a total order on the notes of a piece of music based on their positions on the horizontal axis.

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

- **Vertical**: This refers to the pitch axis on the score. The vertical(pitch) arrangement of the notes determine the emergent harmony when they are perceived simultaneously. Typically, multiple *voices* heard together will lead to an emerging sequence of harmonies, which can be described using chord labels.

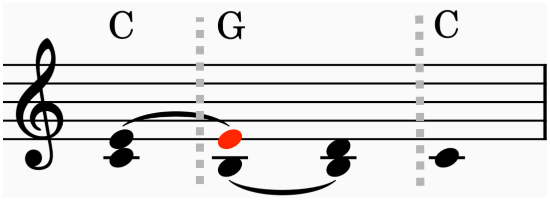


Figure 2.3: Chord labels shown with segment boundaries.

- **Functional**: Functional relations refer to the purpose or function of a note relative to another note. These functions can include repetitions, where both notes have the same *pitch*, or ornaments/neighbour notes, where the child note is a step (single unit of pitch) away from the parent note. These relations can be applied recursively, giving rise to a network of dependencies.

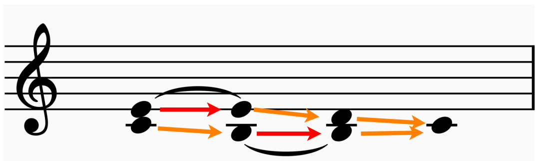


Figure 2.4: Functional dependencies between notes.

2.3.3 Inner Structure

Internally, the protovoice model generates a piece of music through sequential operations on notes, inserting new notes with edges connected to existing notes.

We represent protovoices as a graph G with one vertex for each note, a vertex each for the beginning ($\rtimes$) and end ($\ltimes$) of the piece, and edges that indicate connections between notes. We define a note as $p \in \mathcal{P}$ where $\mathcal{P}$ is the set of pitches, i.e the set of vertical positions on the score[2]. A *protovoice* is a path within this graph.

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.

[2]See Appendix C for a detailed explaination of the pitch representation used in this project

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:

$$p \implies x \to p \quad \text{or} \quad p \implies p \to x \tag{2.12}$$

Operations with two parents are represented by edge replacement.

$$p_1 \to p_2 \qquad \implies \qquad p_1 \to c \to p_2 \tag{2.13}$$

The generation of a piece begins with the empty piece ($\rtimes \to \ltimes$) and involves the recursive application of inner operations, or rules. The full set of rules is left to the appendix, but here are a few as an example:

$$\begin{aligned} x &\implies n \to x & &\texttt{left-neighbor} \\ \rtimes \to \ltimes &\implies \rtimes \to x \to \ltimes & &\texttt{root-note} \\ x \to y &\implies x \to x' \to y & &\texttt{repeat-after'} \end{aligned} \tag{2.14}$$

2.3.4 Outer Structure

The inner structure provided by protovoices captures the sequential and functional organisation of notes, but does not capture when notes are simultaneous. To model simulataneity of notes we introduce *slices*, representing segments of a piece where a group of notes are heard, and *transitions* which contain the protovoice edges between notes in the two neighbouring slices. These provide a higher level abstraction that are used to capture more musical structure.

As slices and transitions contain notes and edges respectively, we call the slices and transitions *outer structure*, and the notes and edges contained therein *inner structure*.

Definition 2.3.1 (Multiset)**.** A *multiset* is a set that allows multiple instances for each of its elements, formally defined as an ordered pair (A, m) where A is the *underlying set* of the multiset, and $m : A \to \mathbb{Z}^+$ gives the *multiplicity*, such that the number of occurences of a in (A, m) is given by $m(a)$.

Definition 2.3.2 (Slice)**.** A *slice* $s \in \mathcal{S}$ is defined as a multiset of pitches $(\mathcal{P}, m)$.

$$s = \left\{p_1^{m(p_1)}, \ldots, p_n^{m(p_n)}\right\} \tag{2.15}$$

Definition 2.3.3 (Transition)**.** A *transition* $t \in \mathcal{T}$ relates two adjacent slices, s_l and s_r, with a configuration of edges e.

$$\mathrm{t} = (\mathrm{s}_l, e, s_r) \tag{2.16}$$

The slices and transitions form a graph given slices as nodes and transitions as edges (containing inner edges). However as transitions only relate sequentially adjacent slices, the outer structure is in fact a *path graph* and can thus be represented as a list of vertices.

Definition 2.3.4 (Path Graph)**.** [3] A *path graph* is a graph which can be represented as an alternating sequence of elements from two sets A and B, defined inductively as:

$$\begin{aligned} P &= a && a \in A \\ P &= abP && a \in A, b \in B \end{aligned} \tag{2.17}$$

Definition 2.3.5 (Outer Structure)**.**
The *outer structure* is thus defined as a path graph, represented as:

$$P = t_1\ s_1\ t_2\ s_2\ \ldots\ t_n \qquad t_i \in \mathcal{T},\ s_i \in \mathcal{S},\ i \in 1 \ldots n \tag{2.18}$$

The outer structure is generated by applying three operations described as production rules recursively:

- A **split** replaces a transition t by inserting a new slice s' and two surrounding transitions t_l and t_r. Each of the edges in t can have by one or more inner operations applied to it. The edges generated by these inner operations can either be discarded, or keep to form the new edges of t_l and t_r.

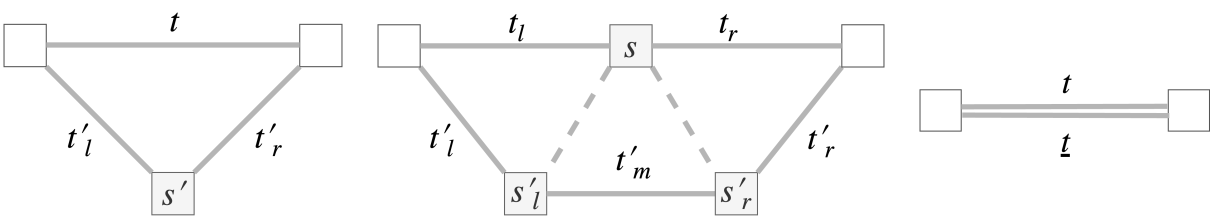
$$t \rightarrow t_l'\ s'\ t_r' \tag{2.19}$$

- A **spread** replaces a slice s by distributing its notes to two child slices s_l' and s_r'.

$$t_l\ s\ t_r \rightarrow t_l'\ s_l'\ t_m'\ s_r'\ t_r' \tag{2.20}$$

- A **freeze** marks a transition as terminal, such that the edges within the transition can no longer have operations applied to it.

$$t \rightarrow \underline{t} \tag{2.21}$$



(a) `split` (b) `spread` (c) `freeze`

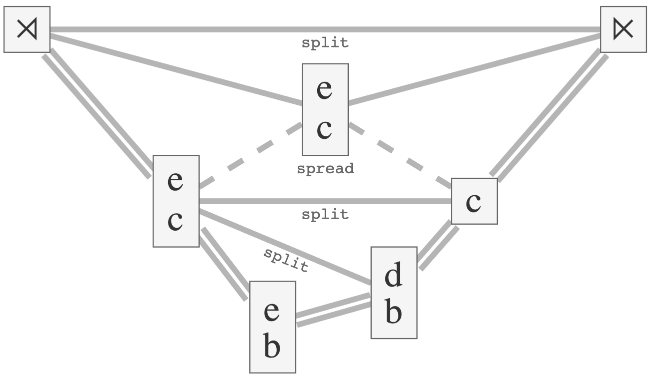
Figure 2.5: The three operations on outer structure.

Figure 2.5 shows a diagrammatic representation of each of these operations. The original slices and transitions are shown at the top of each diagram, while the lower part shows the generated structure after each outer operation is applied.

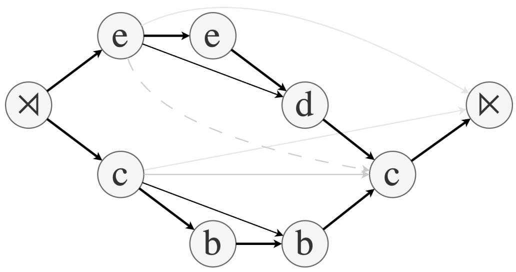
[3] This is refered to as a `Path` in the source code

The generation of a piece thus begins with the empty transition t_0, followed by a `split` operation, which generates the root slice of the piece.

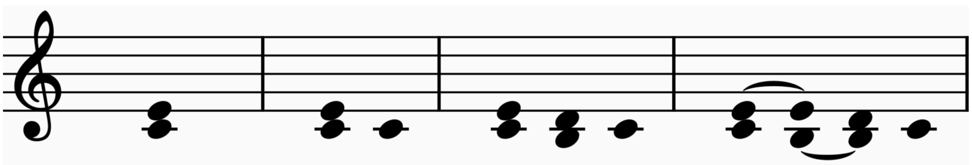
Figure 2.6 gives an example derivation of the short phrase shown in Figure 2.2. In Figure 2.6a, we can see that 4 outer operations have been applied to generate the surface. The spread operation introduces a passing edge between e and c, shown as the dashed line in Figure 2.6b. Note that the vertically aligned notes in Figure 2.6b correspond to the notes in the surface slices in Figure 2.6a, and the final surface shown on the right of Figure 2.6c.



(a) Outer structure



(b) Inner structure



(c) Abstract notation

Figure 2.6: An example derivation of the phrase shown in Figure 2.2.

2.4 Parsing

The goal of parsing is to find a *plausible derivation* of the given surface. For a derivation to be *plausible*, the *structure* it presents needs be consistent with a valid musical *interpretation.* Although this is non-trivial, plausibility can be approximately represented by a probability distribution, analogously to probabilistic context free grammars(PCFGs).

Definition 2.4.1 (Derivation plausibility)**.** The plausibility of a derivation is given by the product of the probabilities of each of the production rules used. Given a derivation D, its plausibility is defined:

$$p(D|\text{surface}) = \prod_{r \in D} p(r|\text{surface}) \tag{2.22}$$

Assuming we can calculate $p(r|\text{surface})$, we can find the most plausible derivation by taking the maximum likelihood estimate (MLE) of the distribution

$$\hat{D} = \arg\max_{D} \left(p(D|\text{surface})\right) \tag{2.23}$$

This presents **two key problems** to be solved:

- **Calculating** $p(r|\text{surface})$. Production probabilities can be viewed as parameters of the model; a common approach with PCFGs is to learn these parameters using machine learning. However, as the protovoice model is not context-free and the volume of data required is not available an alternative method must be found.
- **Combinatorial explosion**: Even if we could calculate $p(D|\text{surface})$ analytically, we would be prohibited by the large branching factor; a single piece can have upwards of $9^{9^{9^{\cdot^{\cdot^{\cdot^{9^9}}}}}}$ possible derivations[4].

2.5 Heuristic Search Algorithms

Heuristic search algorithms are introduced in 1B *Artificial Intelligence*, so I will therefore assume the reader has knowledge of the heuristic search paradigm.

The naive way to solve the above parsing problem is to use an *exhaustive search* strategy. This would theoretically allow us to find the most plausible derivation, but is computationally infeasible.

Best-first search is a heuristic search algorithm that selects the most *promising* node to expand based on a heuristic evaluation function. In general, the heuristic function $h(n)$ depends on the description of n, the description of the goal, information gathered by the search up to that point, and most importantly domain specific knowledge [33].

Beam search is an optimisation of best-first search that serves to reduce its memory requirements by only storing a limited number of best states as candidates to expand, dependent on the *beam width*. Beam search is greedy algorithm, so it does not necessarily produce the optimal solution, but trades optimality for improved complexity.

2.6 Inferring Labels

The task of inferring harmony (ACE) poses **three main challenges**:

- **Segmentation**: Segmentation is splitting of the score into segments which each have an individual label. For example, Figure 2.7 shows each segment separated

[4] I need a better way to represent the scale here. Perhaps I'll try to actually estimate a number

by a dashed grey line. As these segments are typically not given a priori, both the segmentation and harmony needs to be inferred. Performing the joint task of segmentation and labelling is beyond the scope of this project, however, so we will assume that the segmentation is given.

- **Ambiguity**: The notes in a given segment may not be enough to determine the chord label. For example, the slice containing notes D and B in Figure 2.7 could be a realisation of a Bm triad, but the context of the neighbouring slices as well as the functional dependencies of the notes makes it clear that the label should be a G. Furthermore, ambiguity is often used with much license to create artistic interest.

- **Non chord-tones**: The slices in a given segment will typically consist of combination of *chord-tones* and *non chord-tones*. The chord tones directly define the harmony, so to perform ACE, an algorithm will need to distinguish between the two. Previous approaches have involved modelling generation as a noisy process, such that the non chord-tones are considered as noise [43]. The protovoice model allows us *explicitly* explain away non chord-tones. In Figure 2.7c, the non chord-tone is denoted in red. By applying a reduction that removes the neighbour note, E, we result in a single slice in that segment which only contains chord-tones(Figure 2.7d), thus describing the chord label more clearly, shown in Figure 2.7b.

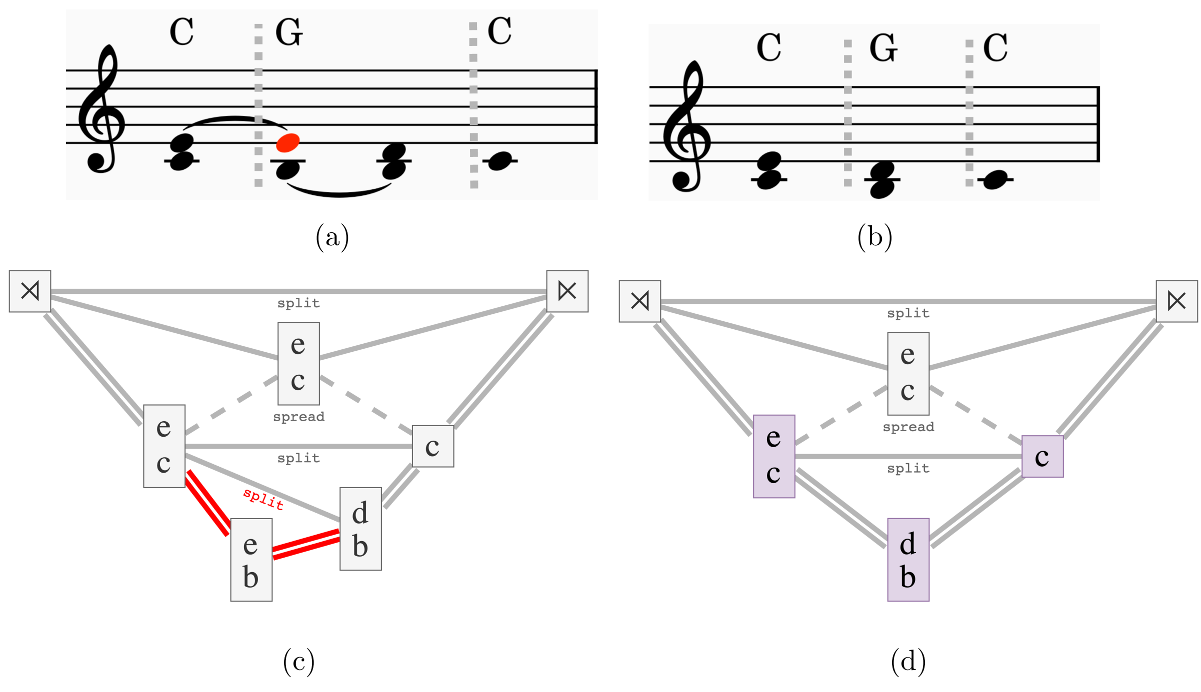


Figure 2.7: Applying a reduction to aid harmonic inference

[5]

[5] I'm considering adding a small section describing some relevant distributions. Dirchelet, Beta, Multinomial, categorical? It took a while for me to get my head around the fact that we used probability distributions of parameters of other probability distributions, e.g. sampling from Dirichlet for parameters of a multinomial or categorical.

2.7 Starting Point

2.7.1 Relevant courses and experience

Haskell I was introduced to Haskell during an internship during the summer before starting this project (July to August 2022). As a result, I had 2 months of experience with the language beforehand. I chose to use Haskell in order to further familiarise myself with the language, and because of its amenability to parsing algorithms. Furthermore, the functional paradigm and rich type system lends itself to modular software development.

Python I have experience coding in Python from personal projects as well as the 1A *Scientific Computing Practical Course*. This was chosen to made use of powerful existing libraries for handling large datasets and running experiments.

IB Two modules from 1B provide a foundation for this project. *Formal Models of Language* introduces the ideas and terminology used in the protovoice model, and *Artificial Intelligence* provides a background for classical search algorithms as well as some of the probabilistic frameworks used in the project.

2.7.2 Existing codebase

This project was built on a fork of the pre-existing *protovoices-haskell* GitHub repository. This repository contains custom data structures and types, allowing interoperability with other projects making use of the same model. I also make use of learned parameters from the implementation of the paper *Bayesian Model of Extended Chord Profiles* [9], as learning these parameters would be beyond the scope of this project.

2.8 Requirements Analysis

Success Criteria

The Success Criteria are given in the Project Proposal. During the preparation phase, I refined the Success Criteria by gaining clarity from the literature related to the project.

This project will be deemed a success given it achieves:

-

Risk Assessment

Table 2.1 shows a list of project deliverables with associated priorities and risk, denoted qualitatively. There is a high general risk in this project due to the fact that the protovoice model is a result of an unpublished thesis, and thus may have flaws that have not been discovered. The highest risk task is designing and implementing the protovoice parser. The heuristic design task is expansive, requiring creativity, research and iterative development, hence the risk is high. The baseline reduction deliverable comprises a standard method for ACE, so there is minimal risk.

Table 2.1: Project Deliverables

ID	Deliverable	Priority	Risk
`core1`	Harmony Model	High	Low
`core2`	End to End Pipeline	High	Medium
`core3`	Protovoice Parser	High	High
`base1`	Baseline Reduction	High	Low
`base2`	Random Search	High	Low
`ext1`	Heuristic Design	Medium	High
`ext2`	Greedy Search	Medium	Medium
`ext3`	Heuristic Search	Low	Very High

2.9 Software Engineering Techniques

2.9.1 Development model

Based on the risk analysis (Table 2.1), I created a plan of which modules to implement in which order, and a list of milestones on a 2 week basis. I used Notion to maintain a list of core tasks and corresponding subtasks with associated priorities, which I used when deciding what to work on. My development strategy drew from the Agile methodology, working on tasks in two-week long sprints, with regular re-evaluations of the plan informed by experimental data and testing.

I made use of GitHub's continuous integration features to run a test suite on the repository after every commit.

2.9.2 Languages, libraries and tools

Table 2.2 shows a justified list of the key languages, libraries and tools used in the project.

Table 2.2: Languages, libraries and tools

Tool	Purpose	Justification	License
Languages			
Haskell	Main language used for the core, baseline and extension implementations	Protovoice model implementation is in Haskell. Functional and amenable to parser development.	GHCL
Python	Secondary language for experiments and analysis	Powerful library ecosystem for running experiments and creating plots	PSFL
Libraries			
Musicology	Haskell Library with data-types for pitches	Contains a robust implementation of spelled pitch classes, which would be tedious to reimplement.	BSD-3.0
Timeit	Lighweight wrapper to show the used CPU time of a monadic computation	This is used to time the runtime of the algorithms as part of analysis	BSD-3.0
Dimcat	Python library: DIgital Musicology Corpus Analysis Toolkit	This library was written to work with the datasets used in this project	GPL-3.0
Numpy	Python library used for preprocessing and analysis	Powerful standard library that is used in conjunction with Seaborn to run analysis and visualise data	BSD-3.0
Pandas	Python library for preprocessing and analysis	This is a standard library for data manipulation and processing	BSD-3.0
Seaborn	Python data visualisation library used for analysis	Creates high quality graphs and charts	BSD-3.0
Tools			
Docker	Containerised software service used to run repeatable experiments	The use of many libraries creates a web of dependencies to be resolved. Ensures the code will last and can be executed on different devices	Free/Paid
Git	Version Control, Continuous Integration	Provides natural backups and allows for reverts to previous commits if necessary	GPL-3.0
GitHub	Hosting source code	Free, reliable hosting	GPL-3.0
GHC	Compiling and profiling.	This is the standard Haskell compiler.	BSD-3.0
Stack	Haskell building and testing	Creates reliable builds, and includes a powerful testing framework.	BSD-3.0
Undotree	Vim Plugin: stores all past actions as a tree	Solves the problem of linear undo history being lost. Protects code between commits.	BSD-3.0
MuseScore	Music notation software	The raw inputs are in the MusicXML format, which is used by MuseScore 3	GPL-3.0
PAT	Protovoice Annotation Tool, Used to view protovoice derivations on a web browser	The protovoice derivations are huge and very complex, so it's vital to have a viewing tool for use in analysis and iterative development	GPL-3.0

2.9.3 Licensing

As shown in Table 2.2, I determined and read up on all of the licensing agreements for the tools used in the project. For the most part, these are all permissive licenses, guaranteeing freedom to use, modify and redistribute as well as permitting proprietary derivative works. The datasets that I use a

2.9.4 Hardware, version control and backup

The code was developed using Vim for Haskell and Visual Studio Code for Python notebook development, on my personal laptop (16' MacBook Pro 2022, M1 Max, 32GB). Experiments were run first locally, then on HPC provided by the EPFL Digital Cognitive Musicology Lab (Dell PowerEdge R740XD Server, 2x Xeon Gold 625R, 768GB), using Jupyter notebooks to conduct the evaluation. I used GitHub for all my notes, development and dissertation writing. Finally, this dissertation was written in Vim with VimTeX.

Implementation

3.1 ProtoVoice Harmony Model

The method I will use to infer harmony using the protovoice model is to find a plausible *reduced surface* S' which has one slice per chord label, having explicitly explained away non chord-tones (ornamentations) through protovoice edges. Given this, the chord labels can be inferred using a model of harmony.

3.2 Repository Overview:

Repository Justification

The repository has been split into four main folders, with the addition of `Main.hs` which serves as an interface between the python experiment code and the algorithms developed in Haskell.

- Firstly, the `src/Core/` folder contains all the core code, including the implementation of the parsing search state and inference functions using the probabilistic model of harmony, as well as some helper code for file handling.
- The `experiments/` folder contains all the python code that is used for this project. The experiments consist of three stages, as described by the three main files: `preprocess.py`, `experiments.py` and `analysis.py`. Splitting these stages up prevents wasteful computation, as all the pre-processing can be done just once, while experiments are run on the processed data iteratively alongside algorithm development.
- The `src/Algorithms/` folder contains all the parsing algorithms including the baseline and extension search algorithms. Having all the algorithms contained in one module allows experiments to be run using any selection of algorithms and input data, facilitating the evaluation process.
- Finally, the `test/` folder contains unit tests and end-to-end tests for use in Continuous Integration.

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/	Running Experiments	115
│ ├─preprocess.ipynb		
│ ├─experiments.ipynb		
│ ├─analysis.ipynb		
│ ├─dcml_params.json		
│ └─inputs/		
└─test/	Unit Tests (Section x)	611

Figure 3.1 illustrates how these modules are connected.

3.3 Core Implementation

The approach taken in the PVHM is as follows:

- First we use the *protovoice model* to *reduce* a musical surface, simplifying the score by removing redundant notes from the surface, using domain specific knowledge to

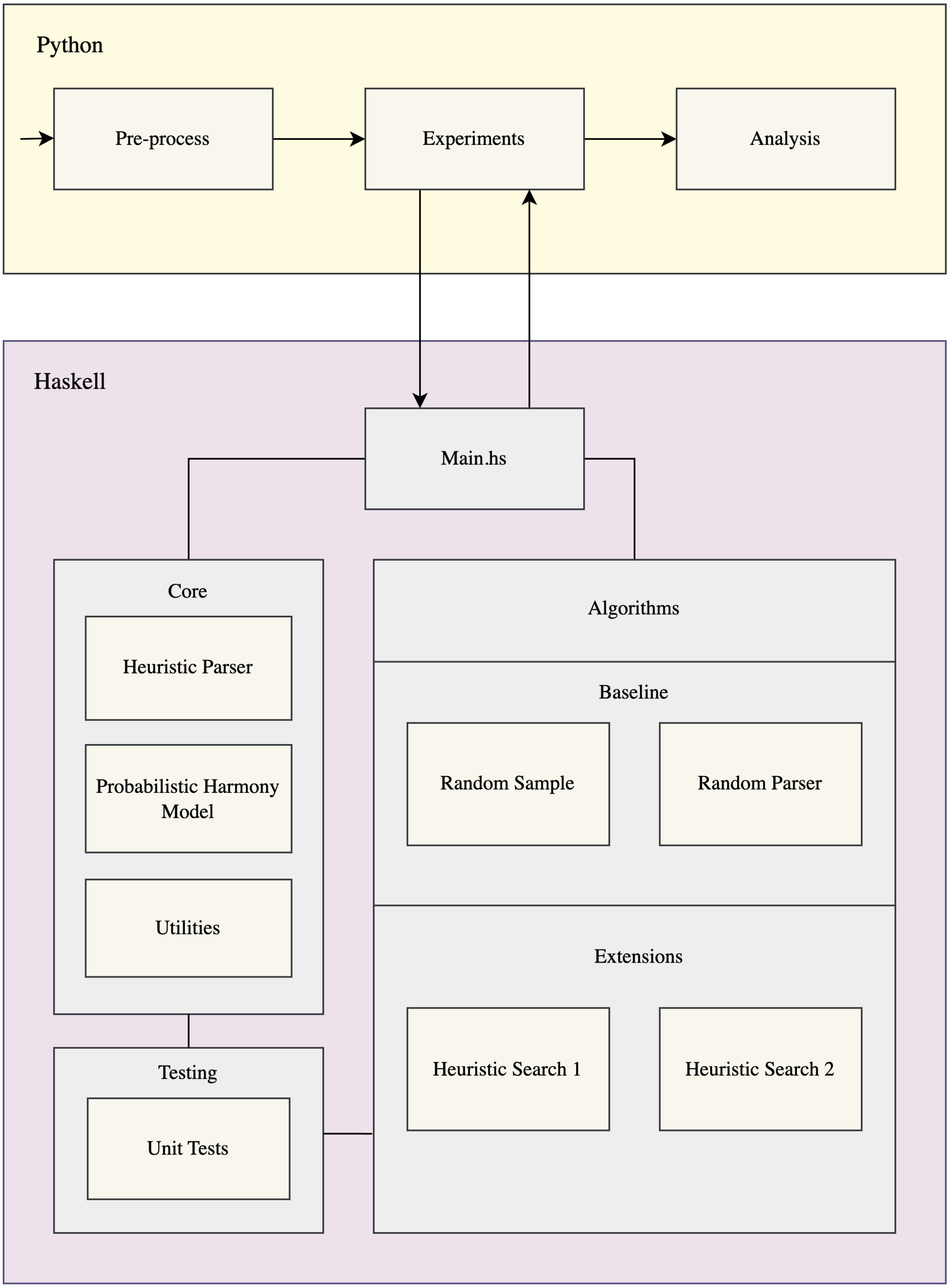


Figure 3.1: Block diagram of project components

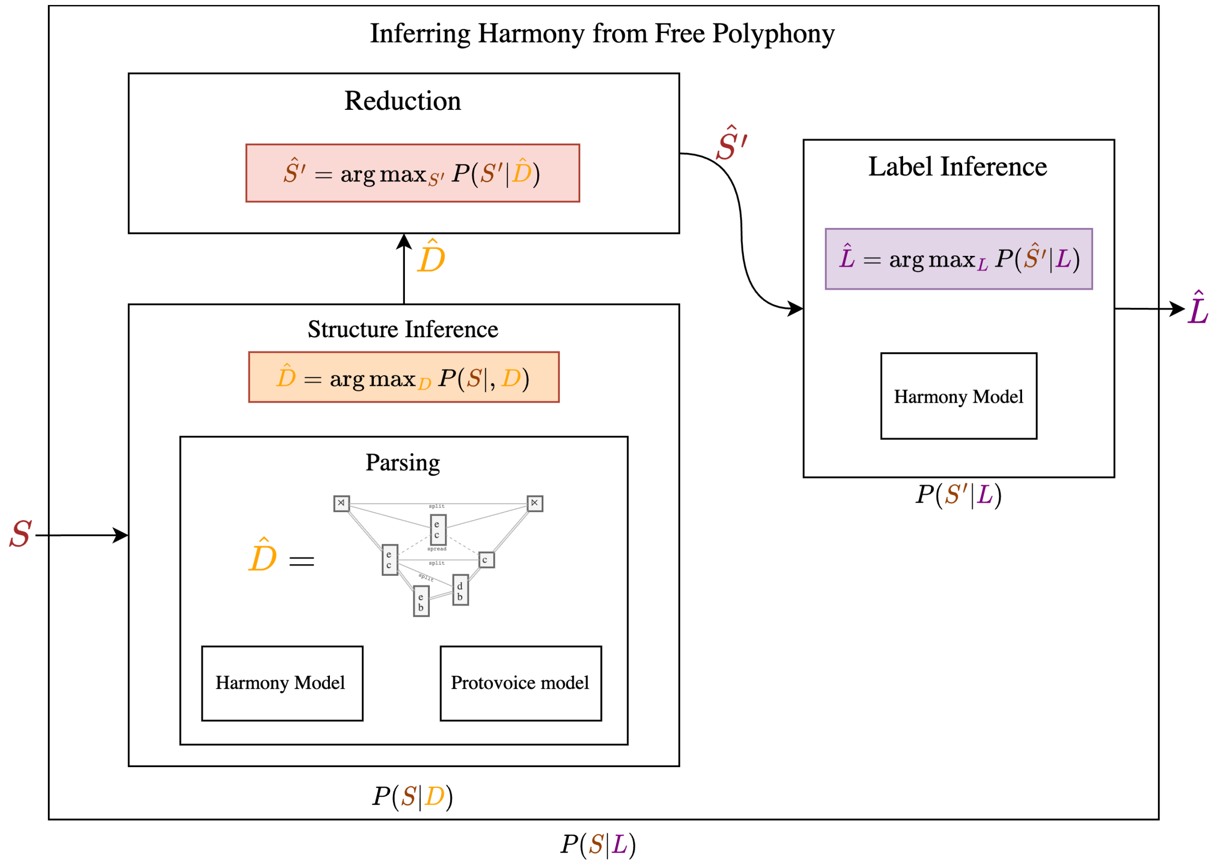


Figure 3.2: Overview of the inference process.

infer the most likely latent structure, $\hat{D}$, which is in turn used to infer the most likely reduced surface, S'.

- Subsequently, using a *harmony model* $P(S'|L)$ we find the conditional likelihood of the *reduced* surface S' given a hypothesised sequence of labels, L.

- Finally, we choose the sequence of labels $\hat{L}$ that *maximises the likelihood* of the reduced score.

3.3.1 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.

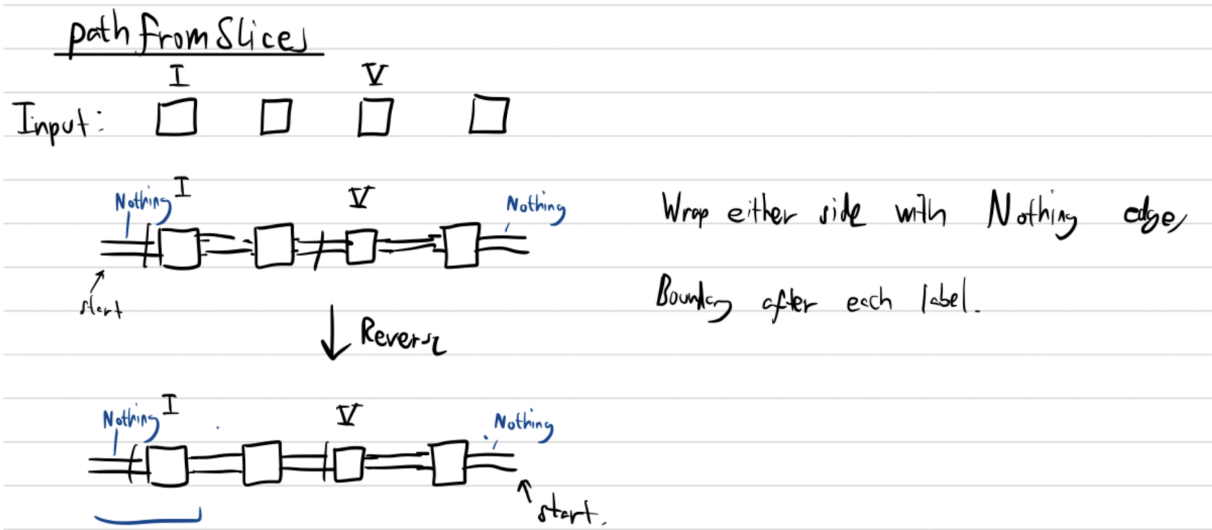


Figure 3.3: Path initiliasation

Definition 3.3.1 (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 reduction by applying 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.

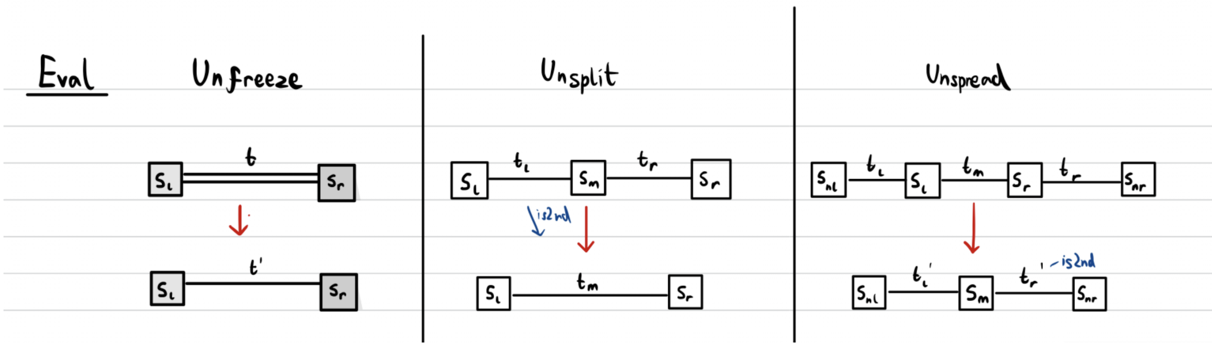


Figure 3.4: Parse operations

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.

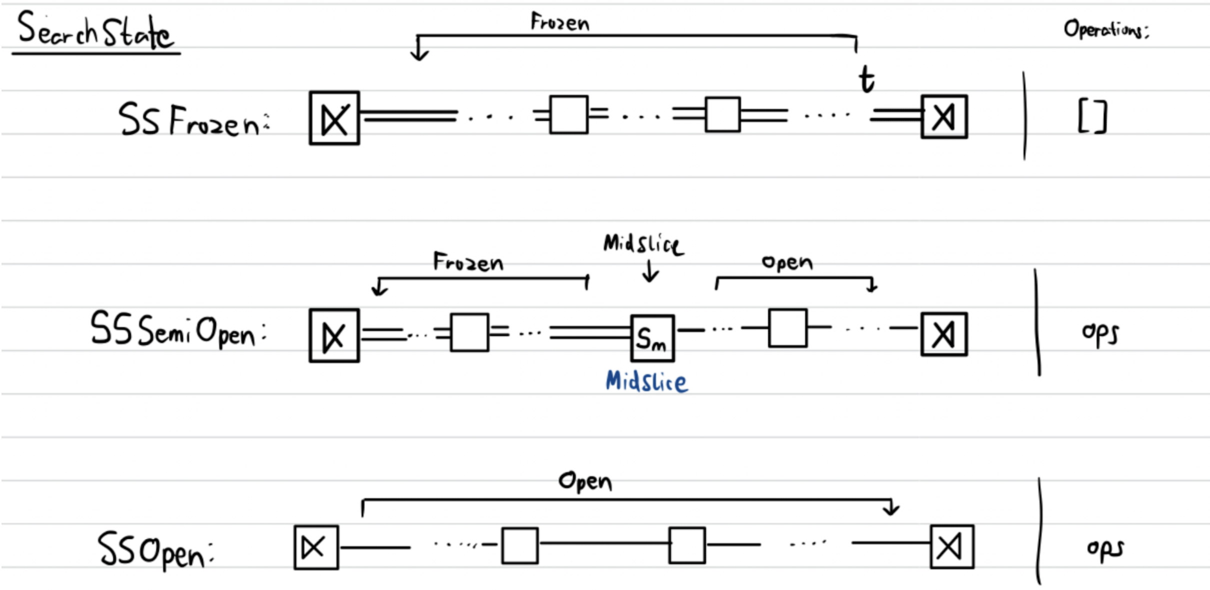


Figure 3.5: Search state

Enumerating State transitions

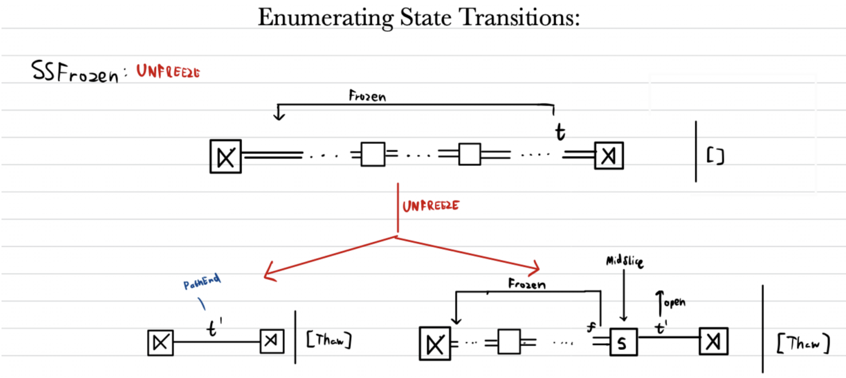


Figure 3.6: Unfreeze operation

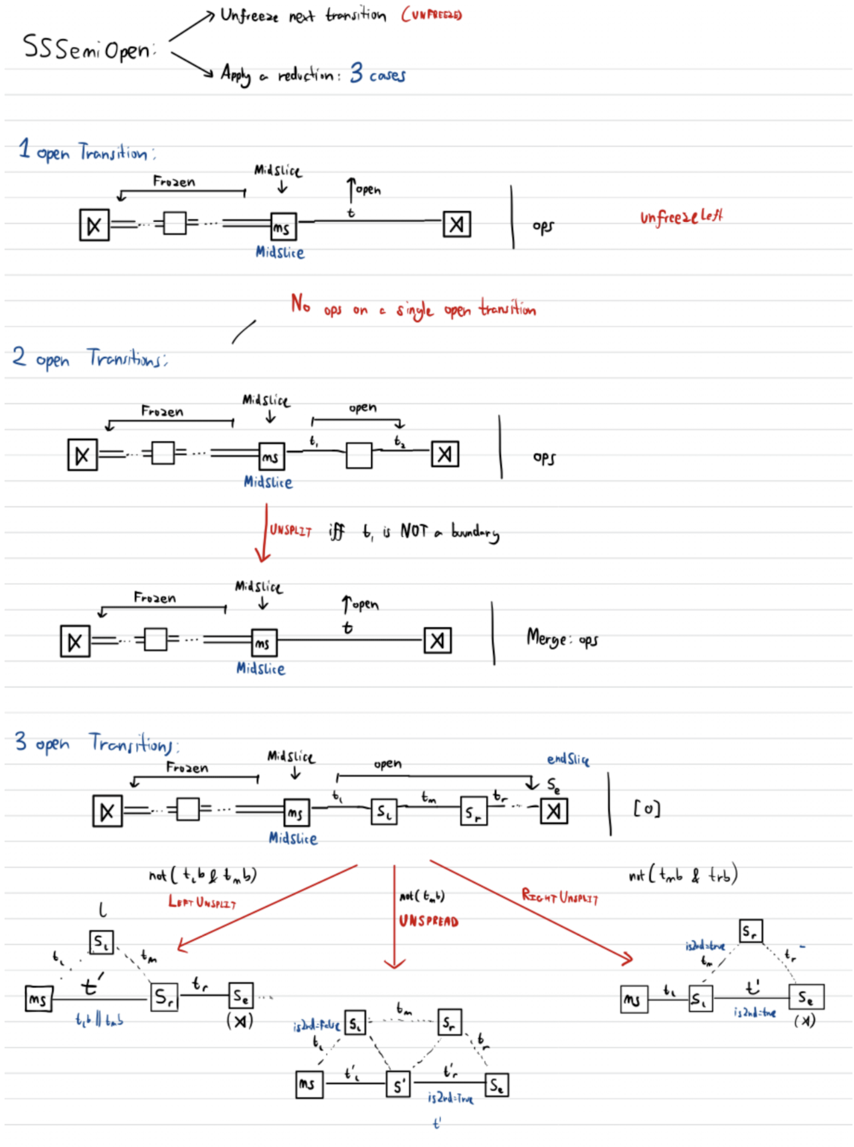


Figure 3.7: Enumeration of operations mid parse. Maybe for appendix? This could be much more concise.

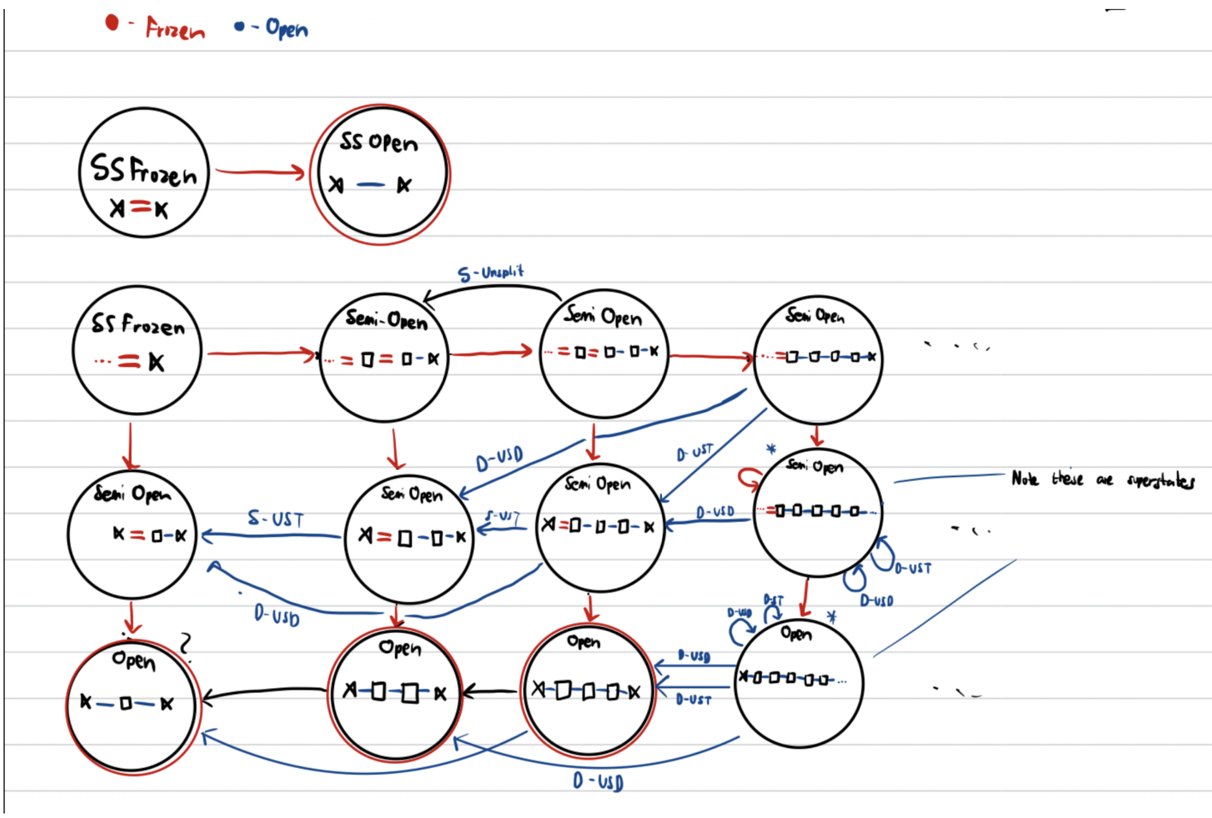


Figure 3.8: State Transition diagram

In the state transition diagram (Figure 3.8), 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.

Could show other maps in the appendix.

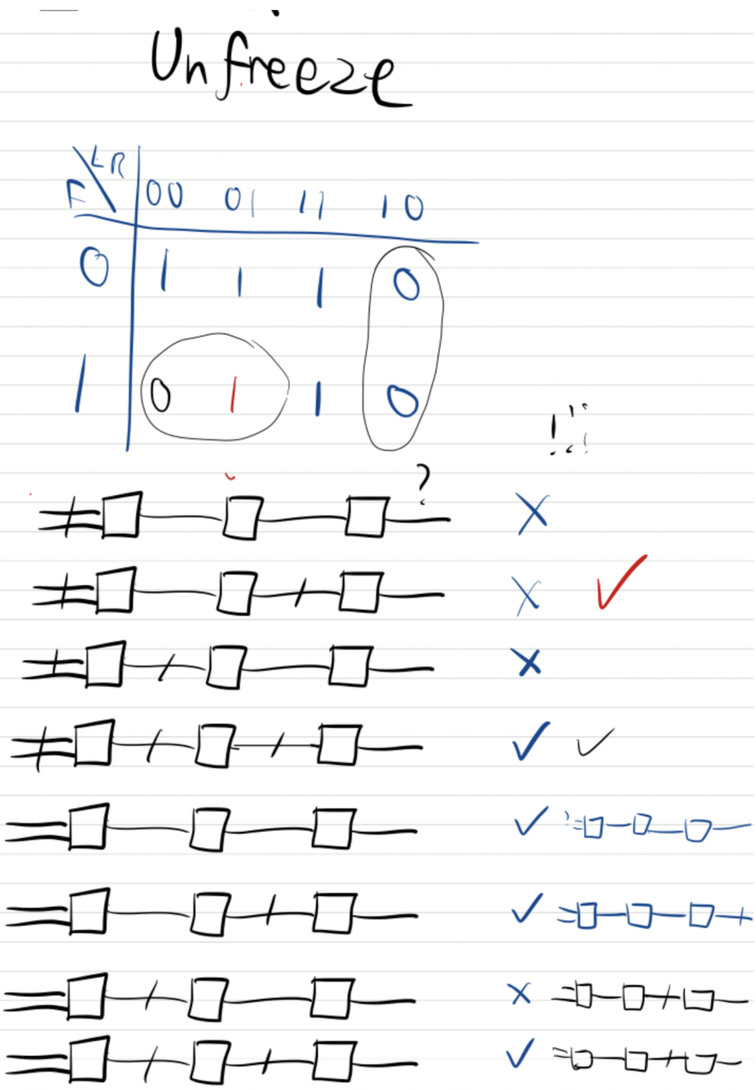


Figure 3.9: Determine boolean boundary expressions for the freeze operations

3.3.2 Probabilistic Model of Harmony

We are making the assumption that the score is a realisation of the latent harmonic entities.

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.

Model definition

We have a set of pitches $\mathcal{P}$ and a set chord-types $\mathcal{C}$. A chord label is defined as a tuple of *root note* pitch and chord-type: $\mathcal{L} = \mathcal{P} \times \mathcal{C}$. We perform a transform of the pitches of each note relative to the chord's root note that is being considered, such that we only need to consider the chord type.

The priors for the model are as follows:

$$
\begin{aligned}
& \vec{\chi} \sim \text{Dirichlet}(0.5, |\mathcal{C}|) && \text{prior of the chord type} \\
& \lambda \sim \text{Gamma}(3, 1) && \text{prior of the note rate} \\
\forall c: \quad & \theta_c \sim \text{Beta}(1, 1) && \text{prior of each note being a chordtone/ ornament} \\
\forall c: \quad & \vec{\phi}^c_{ct} \sim \text{Dirichlet}(0.5, |\mathcal{P}|) && \text{pitch for each ornament} \\
\forall c: \quad & \vec{\phi}^c_{or} \sim \text{Dirichlet}(0.5, |\mathcal{P}|) && \text{pitch for each chord tone}
\end{aligned}
\tag{3.1}
$$

Given this model we use it to generate a single chord as follows:

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

Model inference

Given the model, we use a dataset to learn the parameters of the model. These parameters can then be used for inference as follows. For each datapoint, the chord label

$$
\begin{aligned}
\forall i: \quad & L_i|\vec{\chi} \sim \text{Categorical}(\vec{\chi}) && \text{chord label of each data point} \\
& N_i|L_i, v \sim \text{Multinomial}(v_{L_i}) && \text{notes of each data point}
\end{aligned}
\tag{3.2}
$$

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

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

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

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.

3.3.3 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 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 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 [16]. The `mir_eval` package provides a plethora of metrics to compare chord label predictions [37].

3.4 Baseline implementation

3.4.1 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 [44] to find local keys.

3.4.2 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.

3.5 Extension Implementation

3.5.1 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.

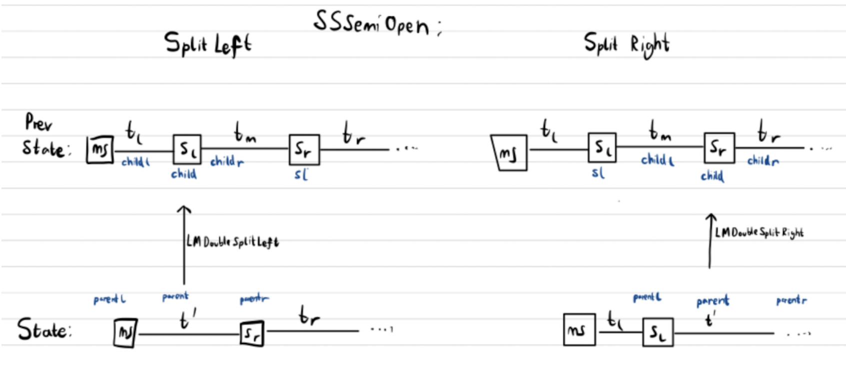


Figure 3.10: Split operation

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.

3.5.2 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

3.6 Testing

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

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 [13]
- So we 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

I'm solving:

$$\hat{L} = \arg\max_{L} \left(p(S|L) \right) \tag{4.1}$$

But I could be solving:

$$\hat{L} = \arg\max_{L} \left(p(S|L)p(L) \right) \tag{4.2}$$

In which case:

To compute the conditional probability $p(L|S)$, we use Bayes' theorem:

$$p(L|S) = \frac{p(S|L)\ p(L)}{P(S)} \tag{4.3}$$

Finding the most likely sequence of labels is found using:

$$\arg\max_{L} \left(\underbrace{p(S|L)}_{\text{likelihood}}\ \underbrace{p(L)}_{\text{prior}} \right) \tag{4.4}$$

The prior probability of a chord sequence $p(L)$ can be learned from a labeled dataset of chord sequences, and the likelihood can be found using a probabilistic harmony model. The likelihood $p(S|L)$ can be found using

This would be better, but was beyond the scope of the project.

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., Ericson, P., Klassmann, S., and Rohrmeier, M. Chord Types and Ornamentation - A Bayesian Model of Extended Chord Profiles. *Open Research Europe* (Apr. 2023).

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

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

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

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

[22] 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.

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

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

[25] 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.

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

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

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

[33] Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley Pub. Co, Reading, Mass, 1984.

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

[35] 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).

[36] 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.

[37] 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).

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

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

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

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

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

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

[44] 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.

[45] 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.

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

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

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

Additional Information

Project Proposal

Inferring Harmony from Free Polyphony

Judah Daniels

April 10, 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 [10]. 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 [10]. 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* [8]. 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 [12], 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* [10] 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 [17], which I will use as part of the data processing pipeline.

ABC – The *Annotated Beethoven Corpus* [30] 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
Mar 3 to Mar 17	Write draft Implementation chapters. In-	Send draft Implemen-

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.

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., Ericson, P., Klassmann, S., and Rohrmeier, M. Chord Types and Ornamentation - A Bayesian Model of Extended Chord Profiles. *Open Research Europe* (Apr. 2023).

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

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

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

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

[22] 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.

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

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

[25] 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.

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

[27] 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.

[28] 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.

[29] MEARNS, L. The Computational Analysis of Harmony in Western Art Music.

[30] 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.

[31] 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.

[32] PARDO, B., AND BIRMINGHAM, W. P. Algorithms for Chordal Analysis. *Computer Music Journal 26*, 2 (June 2002), 27–49.

[33] PEARL, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley Pub. Co, Reading, Mass, 1984.

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

[35] 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).

[36] 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.

[37] 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).

[38] RAPHAEL, C., AND STODDARD, J. Functional Harmonic Analysis Using Probabilistic Models. *Computer Music Journal 28*, 3 (Sept. 2004), 45–52.

[39] ROCHER, T., ROBINE, M., HANNA, P., AND STRANDH, R. Dynamic Chord Analysis for Symbolic Music.

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

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

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

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

[44] 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.

[45] 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.

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

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

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