

Repetition sans Ennui

Human and Algorithmic Discovery of Patterns in Music

I. Ren

The work presented here would not have been possible without ...

© 2021, Iris Yuping Ren
Repetition sans Ennui
Thesis, Utrecht University, The Earth
Illustrated; with bibliographic information

Repetition sans Ennui

Human and Algorithmic Discovery of Patterns in Music

Doctoral Thesis

to obtain the degree of doctor
from Utrecht University
to be defended in public
on Friday 1st January, 2100
at 10:00 hours

by

Iris Yuping Ren

born on Friday 31st May, 1991
in Xicheng, Beijing

Supervisory team: Prof. dr. Remco C. Veltkamp
Prof. dr. Gabriele Keller
Dr. Anja Volk
Dr. Wouter Swierstra

Abstract

Finding "patterns" is one of the common acts in human intellectual endeavours. What does it mean when people say "I see a pattern here"? What does it mean when people point this out in music? We examine the multiple meanings of "pattern" and the methods to find them. Taking music as our focus, we collect, visualise, compare, and exploit human-annotated patterns and algorithmically extracted patterns. In this process, we identify the importance of transformations behind the patterns that repeat and vary. Inspired by functional programming, we use the functional language Haskell to declaratively and compositionally model this connection between pattern and transformation. As the culmination of this work, we explore how to use transformations to relate and classify musical pattern occurrences, which reveals insights for designing pattern discovery algorithms.

Preface

This dissertation presents my research on the discovery of musical patterns at the Department of Information and Computing Sciences at Utrecht University. Before getting into the research, in this preface, I would like to provide some context for the project interspersed with some personal notes. Divided into three parts, I first give a brief introduction to the stages of the evolution of this project, after which I then give a short personal account of the journey that brought me to this project and some thoughts I had along the way, and finally, I finish the preface with a personal view of recent research developments in *Machine Learning (ML)*, *Artificial Intelligence (AI)*, and their relevance to my research.

Evolution of the project

This PhD project began by identifying potential synergies that could be created by combining functional programming and *Music Information Retrieval (MIR)* research. The first step we took was to look into a number of musical pattern discovery algorithms and attempt to combine their power through fusing their output. The discovered patterns turned out to be very disparate according to various types of visualisations, classification methods, and metrics we employed. Around the same time, we also started to consider the importance of and problems with human-annotated patterns and using them as *ground truth*. Then, we started to rethink, model, and examine the musical patterns using Haskell, a quintessentially functional language. We arrived at the idea of using musical transformations to model important aspects of patterns—repetition and variation. After establishing the connections between musical patterns and musical transformations, we continued to use these transformations to analyse algorithmically extracted and human-annotated musical patterns.

This thesis retains some of this order, while certain sections from previous papers are restructured in the hope to creating a more cohesive story. Numerous other aspects influenced and shaped the project, including my background in music, com-

plex system science, and machine learning as well as the rapid development in those areas of research. I will expand on some of them in the following sections.

My journey

This journey has been quite a learning process. A quote from (Cook, 2000) struck a deep chord with me:

"Your academic work told you that music was abstract, transcendent, above the world of napalm and body-bags; the rest of your life told you that music was intimately involved in everyday experience, in the construction and expression of personal and political values."

How I came into this research

(Cook, 2000) goes on saying that "music is one of the means by which we make ourselves who we are". This is certainly true for me, as life before playing music and trying to understand what it all means is blurry to me. I started playing the violin at a young age. From trying to understand something as petty as "why is it difficult to play this note in tune?", to "is this the same melody as before but a bit higher at the end? Why does it sound completely different?", and to the more grandiose and nebulous questions "why am I here and playing this music?" (sometimes in a enjoying sort of way and sometimes with a more negative overtone). Growing up, I also liked the language of mathematics and the consistency and unpredictability from the computers. In the balance between precision and fuzziness, chaos and structure, music gives a unique place to express emotions. Music helped me in affirming and confirming my identity to navigate the ocean of uncertainty that surrounded me.

As obvious as it might seem now, looking back, I still find it outlandish to predict that I would embark on a research project aimed at finding patterns computationally. The directions only concretised more and more when I embarked on several research projects related to music in my master's programmes. After digging into the research area, I was happily shocked to realise that there are many others who have reached for mathematical structure and computational methods to understand music, to find interesting structure and powerful patterns for expressing emotions. This topic was like a kind of meeting point for the mind and the heart. By the time I chose this PhD to combine music information retrieval and functional programming, I was sure that it must be the most interesting topic in the whole intellectual sphere.

Cultures

I was initially drawn to the idea of "music as a universal language". Like many others do, I believe that music may establish a point of connection between cultures. In a collision of cultures, it is not uncommon for people to go through the process of "gaining some understanding of the cultural other, but also of shifting your own position, constructing and reconstructing your own identity in the process" (Cook, 2000). What relates to this is perhaps the fact that my ethnic origins (Chinese) has little to do with the music I use for this project (mainly not Chinese). Although music in the West is quite distinct from the rest of the world, it is not unfamiliar to me, as the trend of learning Western music and instruments was prevalent in China when I was a child.

In terms of my taste in music generally, I'm a proud audio omnivore. I enjoy all kinds of simple and complex music from various cultures. I gradually came to know more about the different layers of subtleties in the history and the resulting space between music of the West and the rest: how they are created, heard, and described. It is perhaps exactly because of these diversities and discrepancies that I developed a curiosity in the objective and quantitative aspects of music.

A few interesting moments

In the next few paragraphs, using one paragraph each, I will describe a few enjoyable moments I had while working on this dissertation in the past years.

When listening to music, I pay more attention to the various states of mind I enter, switching swiftly between the foreground and background of consciousness. For example, this might happen when I use music functionally to focus and meditate, or when I perform music as written on a piece of sheet music, or when I just spontaneously improvise and be at the receiving end of whatever sensations that comes to me. I realised that very few people actually note down the musical patterns they experience or create, and they come and go so fast.

One of my favourite objects are wind chimes. They acquaint me closer with the concept of randomness and patterns. Imagine a wind chime of the tones G - A - B - D - A - G - B - D, which might produce a pattern of GBDGBAGBD by wind one day. It also reminds me of number patterns such as 123234345, or 1002000300, which can very well be found in random strings, while it's easy to discern some patterns from them. In contrast to the randomness, I can also "play" the wind chimes and control it to a certain extent: the boundaries begin to blur when randomness gets produced by

deterministic process. The full cycle was completed when I re-realised determinism can also be generated from randomness, e.g. Gaussian distributions.

After one discussion with my supervisory team, I became fascinated with one of the simplest type of data—sinusoid. There are so many repetitions and structures in this one type of shape. Combining with what I've already know of Fourier transformation, I felt amazed again to realise that these simple repetitive shapes can be the building blocks in approximating so many other complicated shapes.

I started to realise more and more often that in everyday life, we have many loops, repetitions, patterns, structures, and routines all around us. Much of the time, we actually repeat without much thought. We do not think of every step, the left and right, we take while we walk; we do not think of the repeated motion we use to brush our teeth or make a stir-fry. Periodicities, or "rhythms", such as these, are prevalent on all scales. We do not need to think of them all the time, otherwise it is easy to imagine how boring it would become; nevertheless, by recognising these repetitions and making small changes, one can sometimes achieve greater meaning in life.

COVID and RSI

At a late stage of this project, COVID hit the world. For me, Repetitive Strain Injury (RSI) has become an extra companion around the same time. They influenced many aspects of my life, and made me think of some algorithmic equivalences and problems. Vaccines, for example, are synthetic attacks on the immune system, and using synthetic data (we will do this in Chapter 4), from this angle, can perhaps be viewed as vaccines for the algorithms. Algorithms helped my RSI to a certain extent, in terms of speech recognition for voice typing and facial recognition for pointer actions. Using "smiles" to perform single click works very accurately, except when partially occluded by the microphone, and perhaps sometimes to a fault—when in online meetings, I'm always worried that I'll click myself out of the meeting before I am supposed to. Speech recognition, by contrast, is still comically discombobulated sometimes: Haskell is either "high school" or "Pascal", MIR is "Am I are", "Bach" is "bath"; sometimes with a little accent of mine, "algorithms" is "organisms", and "excerpts" is "accept". Other frustrations include the limited time-window in which I must speak, various punctuation that takes much more effort to voice type, and the difficulty of modifying whatever has already come out of my mouth. These are the difficult problems of algorithms that we sometimes see in musical pattern discovery as well: ambiguity, similarity, less ergonomic settings, limited memory, and a lack of ability to recognise and manipulate structures. With new innovations such

as screen recognition, code typing systems, wearable keyboards, and virtual reality, I'm optimistic that technology will continue to change our lives.

Closing the journey

During the process of this PhD, I draw inspiration from many sources for studying musical patterns, such as literature from various subjects, people around me, everyday life, and vice versa. I believe I will revisit this experience in memory many times for the rest of my life, for a variety of reasons, one of which being that there are other experiences I cannot recount in this preface. In this dissertation, I would like to share what I learned about musical patterns, especially with help from functional programming.

This time in research

The current research climate has somewhat impacted me. From two different communities, music information retrieval and functional programming, we see growing connections with the machine learning community and all the associated research fields.

Different styles of research, such as fast and slow science, is something I have learned along the way, too. In the fast developing field of [ML](#), there are sometimes 200+ ML tools made available for one task and 6000 papers published in one sub-area in the past few years. From simple algorithms to the more complex machine learning and black box models, they are only a small part of the rapidly evolving technologies that can be applied to music research.

In broader terms, what has piqued many people's interest are topics such as bias in human data, software glitches, and deep fakes. Leaving computer science, we see many other issues such as mental health, climate change, and the COVID pandemic. All of them, though, seem to can be linked to [AI](#), which is one of the hottest topics in today's research landscape.

From developmental psychologist Howard Gardner, we know that intelligence comes in many forms: linguistic intelligence, musical intelligence, logical-mathematical intelligence, spatial intelligence, bodily-kinesthetic intelligence, intra-personal intelligence, interpersonal intelligence. In addition, intelligence has also been categorised in a different direction: general intelligence, broad cognitive abilities, task-specific skills. Each of these domain-specific intelligences can be used together with "cognitive fluidity", whereby we combine them into an integrated

meta-intelligence (Mithen, 1996). One way or the other, *Artificial General Intelligence (AGI)* seems to be grand ambition for many. Music, as a means of expression that is intrinsic human perception and cognition, should attempt to be accounted for by any computational system hoping to comprehend intelligence as a whole.

There are, of course, arguments that support the independence of music intelligence from other types of intelligence. Steven Pinker has famously called it "auditory cheesecake", emphasising music's position as a by-product, and not as essential. Counter theories emphasise the fact that music could be essential for humans in the evolutionary process for social bonding and credible signalling, and argue that music is not optional, but bundled with the hardware of being human. Whatever the case might be, one thing is certain: musical intelligence is something human.

If a machine would exhibit this intelligence, how would it look? All this software and these algorithms we have at the moment, are they the final form of human thought? Would this ideal necessarily require big data, or the opposite, small and detailed data, or a combination of both, data that is both big and detailed? The same can be said about the size of models: how many parameters are too many and what are the environmental costs incurred by training such models? The community is still looking for answers.

Edsger Wybe Dijkstra famously commented that "The question of whether a computer can think is no more interesting than the question of whether a submarine can swim." As it stands now, many contend that **AI** is a misnomer, it should be instead artificial cleverness, and sometimes we are more worried for artificial stupidity rather than creating intelligence. The correctness of these statements are constantly in flux because of the rapid development of all types of models.

Just to test out where we are with all the progress, below I've included a reinterpretation of this PhD project generated from an *Natural Language Processing (NLP)* model. The model was pre-trained and tries to rewrite some bullet points I've given it.

Modelling of artistic activities requires methods that bridge between disciplines. Musical acquisition, perception, production, composition, and appreciation has much in common with other areas like language, programming, and mathematics. Thereby, the realization of a common research environment has the potential to increase research quality and volume. Furthermore, the success of pattern recognition software have brought about interest in applying these same methods to real world problems. The aim of this project is to develop a system that extracts higher level cognitive structures from musical data. The project also

aims to build an integrative framework that combines computational, cognitive, and empirical knowledge about music.

Although the algorithm did not get the spelling conventions right and trespassed into research scopes I'd have stayed out of, it generated original text based on original input. Moreover, I did make some edits (< 10 words), but when I was editing it, I liked the fact that I would never have written this text myself, and I would love to have it to help with my writer's block. Would the computer be able to do as well when it comes to performing musical analysis? Hybrid intelligence, such as cyborg chess or centaur chess introduced by Garry Kasparov, may be part of the answer. With this, I'll conclude the preface and move on to the main text, where I'll expand further on musical pattern discovery algorithms.

Acknowledgements

List of Acronyms

AGI	Artificial General Intelligence	xii
AI	Artificial Intelligence	vii, xi, xii, 10
ANOMIC	ANnOate MusIC, an annotation tool we devised ..	73, 76, 77
CC	Comparative Classification, a method we devised	20, 22, 109, 146, 222
CNN	Convolutional Neural Network	228
COSIATEC	Data COmpression using SIATEC	59, 60, 143
DL	Deep learning	12, 64
DNN	Deep Neural Network	64
DSL	Domain Specific Language	21, 24, 103, 153, 179
FP	Functional Programming	7, 23, 151, 153, 155
GBM	Gradient Boosting Machine	131
GHC	Glasgow Haskell Compiler	153
GTTM	Generative Theory of Tonal Music	44–46
HCI	Human-Computer Interaction	10, 110
HEMAN	Human Estimations of Musically Agreeing Notes ..	76, 77
IOI	Inter-Onset Interval	182
JAMS	JSON Annotated Music Specification	20
JKU-PDD	The Johannes Kepler University Patterns Development Database	111, 203, 235

List of Acronyms

kPCA	kernel Principal Component Analysis	215
LDA	Linear Discriminant Analysis	131
LVQ	Linear Vector Quantisation	131
MDL	Minimum Description Length	65
MDS	Multidimensional Scaling	64, 118, 213
MGDP	Maximally General Distinctive Pattern	61
MIDI	Musical Instrument Digital Interface	15, 121, 184
MIR	Music Information Retrieval	vii, 10–12, 25, 39, 62, 66, 74, 110, 153, 235
MIREX	The Music Information Retrieval Evaluation eXchange	12, 33, 46, 47, 51, 53, 74, 111, 117, 204, 235
ML	Machine Learning	vii, xi, 25, 64, 103, 110, 230, 258
MTC-ANN	The Meertens Tune Collections: The Annotated Corpus	17, 111, 118, 119, 121, 124, 129, 138, 203, 236
MTP	Maximal Translatable Pattern	59, 60
NB	Naive Bayes	131, 132
NLP	Natural Language Processing	xii, 2, 65, 66
OED	Oxford English Dictionary	32, 158
OOP	Object-oriented Programming	37
PAF	Pattern Annotation Framework, an annotation tool we devised	73, 76, 77, 81
PatMinr	Pattern Miner	62, 113
PCA	Principal Component Analysis ..	11, 12, 118, 121, 124, 129, 138
PL	Programming Language	153, 174
PLP	Programming Language Processing	174
PP	Pattern Polling, a method we devised	11, 20, 22, 109, 127, 146, 222
RF	Random Forest	131
RNN	Recurrent Neural Network	64, 228

SIA	Structure Induction Algorithm xix, 59–61
SIACF1	SIATECCompress-F1 59
SIACFP	<i>Structure Induction Algorithm (SIA)</i> for r superdiagonals and Compactness Trawler-Categorisation and Fingerprinting 59, 61, 117, 142
SIACP	SIATECCompress-Precision 59
SIACR	SIATECCompress-Recall 59, 124
SIAR	SIA with a sliding window of size r 59, 61
SIATEC	Structure Induction Algorithm for <i>Translational Equivalence Class (TECs)</i> Meredith <i>et al.</i> , 2001 ... 59, 60
SotA	State-of-the-Art 22, 57, 74, 113, 114, 130, 150
SVM	Support Vector Machine 131
SYMCHM	Symbolic Compositional Hierarchical Model 58, 124, 138
TEC	Translational Equivalence Class xix, 59, 60
tSNE	t-distributed Stochastic Neighbor Embedding 215
VM1	Algorithm name 58
VM2	Algorithm name 58

Contents

List of Acronyms	xvii
1 Introduction	1
2 Pattern and Pattern Discovery	25
3 Gathering Human-Annotated Musical Patterns	73
4 Comparisons and Evaluation of Musical Pattern Discovery Algorithms	109
5 Computation, Functional Programming, and Music	151
6 Modelling Patterns with Transformations using Haskell	177
7 Using Transformations to Understand Patterns	197
8 Conclusions and Future Work	221
Appendices	233
Appendix A Datasets	235
Appendix B Introduction to Haskell	239
Appendix C Introduction to Category Theory	245
Appendix D Pattrans Package	249
Appendix E List of Publications	255
Glossary	257
Samenvatting	261
Curriculum Vitae	263

CONTENTS

References

265

Contents

List of Acronyms	xvii
1 Introduction	1
1.1 Overview	1
1.2 Challenges	9
1.3 Approaches	10
1.4 Disciplinary contexts: a summary	11
1.4.1 Music Information Retrieval and pattern discovery	12
1.4.2 Musicology, music theory, psychology, and cognition	12
1.4.3 Functional programming, pattern, and transformation	13
1.4.4 Broader related areas and connections	14
1.5 Scope	15
1.5.1 Why monophonic	15
1.5.2 Why symbolic	15
1.5.3 Datasets and generalisability	16
1.5.4 Format of patterns	16
1.5.5 Inter- Intra- opus, patterns, and occurrences	17
1.6 Thesis statement	18
1.7 Dissertation outline	19
1.8 Contribution	21
1.9 Notes on reading this dissertation	22
2 Pattern and Pattern Discovery	25
2.1 Overview	25
2.2 Pattern as a concept	31
2.3 Musical patterns and related keywords	38
2.3.1 In MIR and other algorithms	38
2.3.2 More music-specific	46
2.4 Pattern discovery algorithms	57
2.4.1 Musical pattern discovery algorithms	58
2.4.2 Generic pattern discovery	63

Contents

2.4.3	Pattern discovery in other domains	65
2.5	Our working definitions	67
2.6	Summary	68
2.7	Discussion	69
3	Gathering Human-Annotated Musical Patterns	73
3.1	Overview	74
3.2	Setup for HEMAN, PAF, and ANOMIC	77
3.2.1	Music material	77
3.2.2	HEMAN	78
3.2.3	PAF	81
3.2.4	ANOMIC	84
3.2.5	Differences between the tools	86
3.2.6	Differences between the experiments	87
3.3	Our methodology and other metrics	89
3.3.1	Agreement analysis	90
3.3.2	Feature analysis	91
3.4	Results	94
3.4.1	Agreement analysis	94
3.4.2	Feature analysis	96
3.5	Conclusion	100
3.6	Summary	102
3.7	Discussion	102
4	Comparisons and Evaluation of Musical Pattern Discovery Algorithms	109
4.1	Overview	110
4.2	Our approaches	114
4.2.1	MIREX metrics	114
4.2.2	Visual comparison	115
4.2.3	Combining the algorithms	125
4.2.4	Classification	129
4.2.5	Synthetic structure and data	138
4.2.6	The ground truth and single score problem	145
4.3	Summary	146
4.4	Discussion	147
5	Computation, Functional Programming, and Music	151
5.1	Overview	151
5.2	Functional programming language Haskell and music	155

5.3	Approach musical pattern discovery using functional programming: transformations	158
5.3.1	Our proposal: Pattern and transformation	158
5.3.2	Musical transformations	159
5.3.3	Implicit and explicit transformation in musical pattern research and algorithms	165
5.3.4	Generic transformations	168
5.3.5	Perception	170
5.3.6	Transformation and Similarity	172
5.4	Summary	173
5.5	Discussion	173
6	Modelling Patterns with Transformations using Haskell	177
6.1	Overview	177
6.2	Musical transformations in Pattrans	180
6.3	Pattrans	184
6.3.1	Basic types	184
6.3.2	Transformation checking	186
6.3.3	Compositions of checking transformation	187
6.3.4	An excursion into category theory	189
6.3.5	Approximation	191
6.4	Summary	194
6.5	Discussion	194
7	Using Transformations to Understand Patterns	197
7.1	Overview	197
7.2	Querying for patterns	199
7.3	Analysing transformation data	201
7.3.1	Data and background	203
7.3.2	Transformations in human annotations	205
7.3.3	Transformations in algorithmic output	206
7.3.4	Comparing patterns across different corpora	208
7.3.5	Statistical analysis	210
7.3.6	A transformation profile for each pattern and PCA	212
7.4	Summary	215
7.5	Discussion	217
8	Conclusions and Future Work	221
8.1	Summary of the chapters	221

Contents

8.2 Applications	222
8.2.1 Data collection	223
8.2.2 Patterns for Education	223
8.2.3 Creativity and algorithmic composition	223
8.2.4 Music, health, and culture	225
8.3 Looking Back	225
8.4 Looking Ahead	229
Appendices	233
Appendix A Datasets	235
A.1 JKU-PDD	235
A.2 MTC-ANN	236
A.3 HEMAN and its different versions	237
A.4 Other datasets	237
Appendix B Introduction to Haskell	239
B.1 Defining type, type class	239
B.2 Function, pattern matching, lambda expression, currying	240
B.3 Currying and higher-order function	240
B.4 List comprehensions and laziness	240
B.5 Recursion	241
B.6 Functor and more	241
Appendix C Introduction to Category Theory	245
Appendix D Pattrans Package	249
D.1 Basic types and functions	249
D.2 Approximation	250
D.3 Query	252
Appendix E List of Publications	255
Glossary	257
Samenvatting	261
Curriculum Vitae	263
References	265

Chapter 1 Introduction

De patronen in muziek en alle andere kunstvormen zijn de sleutels tot het verwerven van kennis.

– Plato (translated to Dutch by [Scherder](#) (2017))

Are we, as a species, united in how we perceive, understand, and create patterns? Is pattern finding a mechanical process that can be replicated using machines? These are grand questions that a single PhD dissertation could hardly be expected to address comprehensively. This dissertation's aims are, therefore, more modest: motivated by these deeper questions, we strive to provide evidence, analysis, and tools to address these questions in the domain of musical pattern discovery, using a variety of algorithms and symbolic, monophonic data.

In this very first chapter, we start with a broad overview of the topics and challenges covered in this thesis. Following that, we will look at the approaches we take to address these challenges. Expanding on these approaches, motivations, and challenges, we present brief summaries of the disciplinary contexts to contextualise the research. The next subsection highlights the thesis statement of this dissertation, followed by the specific scope we work within. What follows is a chapter-by-chapter outline and an overview of our contributions. Finally, we provide some notes on reading this dissertation.

1.1 Overview

Pattern

Over millennia, human civilisations in their creativity have found patterns that are pleasing, repetitive, useful, meaningless, abundant, scarce, amongst many other attributes. It is not difficult to notice that they appear in a variety of domains and that "The use of the word 'pattern' is ubiquitous in written and spoken discourse." ([Toussaint, Toussaint, et al., 2014](#)). Music is not an exception: "Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern."

1 Introduction

(Whitehead & Price, 2001), just to quote one amongst many who have written about patterns in music.

Pattern is a central concept for this dissertation. Chapter 2 will expand further on this concept.

Past and Present

The past few years have seen a rapid rise in the number of pattern-finding algorithms. Thanks to the growth in AI, machine learning, deep learning research, domains such as computer vision and [Natural Language Processing \(NLP\)](#) have made significant strides. Music, which is rich in its diversity, structure, and cultural and artistic values, is also becoming more automated in its creation and analysis. With these developments, this dissertation focuses on algorithms that can discover patterns in music.

While it might sound odd to some, the convergence of computational methods, pattern finding, and music research is a rather natural and old one, and so is the connection between humans and machines in general. Machines have become an irreplaceable tool for the survival and prosperity of us *Homo sapiens*. For centuries, algorithms have become a part of our everyday lives (McLean and Harlizius-Klück, 2018). The desire to come closer to fathoming our mental capacity as well as compare ourselves with the mechanical process can be traced at least as far as Lovelace. Since then, scientific approaches have been widely adopted across a wide range of disciplines.

Human and Machine

Humans and machines tend to find different patterns in different ways. For a human to recognise patterns, a number of cognitive capabilities are usually necessary: the ability to conceptualise and keep the patterns in their minds, to recognise new developments, and maintain this goal of finding patterns throughout the process. Some call it "mental time travel", as it demands the use of one's imagination to create a vision of a hypothetical future (Lent, 2017). Some other patterns can be found without much attention and focusing, such as recognising cats and dogs, or an exact repetition of a few notes. A diverse set of patterns may be discovered because "human mental processes are noisy, leaky, graded..." (Margolis, 1987).

Machines, conversely, typically run algorithms on a von Neumann architecture, are extremely precise (to a fault, sometimes) and fast. Unlike the human brains that integrate computation and memory with the same underlying structures of neurons,

the von Neumann architecture has separate processing and memory units, which also tend to be monolithic ones, different to humans (Davies, 2021). With a whole host of other differences (Von Neumann *et al.*, 1958), such as energy efficiency, data efficiency, sociality, and complexity, the patterns found by machinic algorithms can be perplexing and divergent from the patterns perceived by humans. For example, we often see the complexity of algorithmic output exhibited in the sheer number of the "patterns" discovered by machines.

Pattern recognition research is bridging the gaps with methods such as neural networks, Bayesian methods, and symbolic reasoning. The gradually increasing resemblance, as well as some persistent dissimilarity of machine intelligence to that of humans, prompts comparisons between the two. This comparison, given the diversity of methods in pattern discovery, is not an easy task, particularly when there is a lack of specification, which is often the case. A precise division between what is recognised as a pattern and what is not is rare—even we humans disagree with each other on different schools of thought and tendencies regarding what we see as a pattern. When there are multiple algorithmic results and multiple human judgements, a careful comparison would be needed. However, it also offers opportunities: it is a perfect testing environment for the generalisability of the algorithms.

Music

These subjective differences in perceiving patterns are even more prominent in music than in other areas. Music has multiple roles to play in our lives, such as providing entertainment, defining who we are, inspiring, evoking memories and emotions, or simply in the background and going unnoticed. What we think of as a musical pattern may vary due to different social, historical, biological, and mathematical influences.

Nevertheless, these aspects of musical patterns also inspire many scientists and programmers. We can take music as a vehicle to construe models of how we think about abstractions and how much of it can be modelled by computation. The many nuances of musical patterns may offer fresh perspectives. By drawing connections and analogies with music, an array of human-friendly insights that are not directly available to intuition can be gained, too. We will not be able to see all the potential benefits in this first chapter, but we will provide some examples and a panoramic view of the dissertation next.

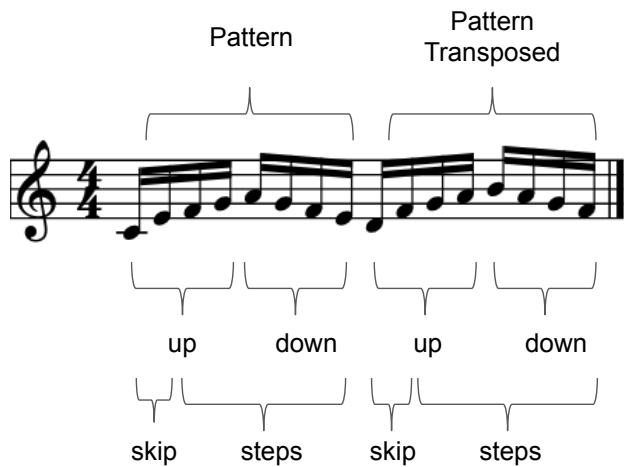


Figure 1.1: Different patterns in a piano étude. We can identify a transposition and the two other possibilities as bracketed under the stave as patterns: a combination of repeated upward/downward motions or different intervallic combinations.

Patterns in music: some examples

It has been much more than a century ago since music scholars annotated musical scores and started to use terms such as motif, theme, unit, and pattern to refer to annotated note collections. We show three examples, to give a more intuitive and concrete idea of patterns that can be seen in music. Figure 1.1 shows the first few notes of a piano étude. There are a few ways that these few notes can be grouped into patterns: grouping the first eight notes, and the rest are these eight notes transposed; grouping together the ups and the downs; grouping together the skips and the steps. Even with a simple few notes, it is nontrivial to describe all the patterns that can be discerned.



Figure 1.2: Different patterns in Mozart's Adagio K.331. This example shows how ambiguity in music can be performed and perceived subjectively (Lerdahl and Jackendoff, 1985).

A more musically interesting example is perhaps the two bars shown in Figure 1.2. This example shows how listeners can have two different interpretations of what the musical patterns are. In fact, this example has been intensively studied: (Gabrielsson, 1987) identified that this piece had been published in different editions with different phrasings. As part of this research, piano performances of the piece with differing interpretations that correspond to these different phrasings were analysed. This is a prime example of *ambiguity* in music. One can either accept both views or choose one side over the other.

In Figure 1.3, including again the piano étude, we describe a few more examples in plain language and in musical terms. Without relying on specific musical terms, we can describe how these pattern occurrences relate to each other as: in (a), the second occurrence is based on the first but higher; in (b1), the second line of melody is almost an exact repetition of the first line, but with a slight change in the middle; in (b2), the entire melody is almost an exact repetition of (b1), but with slight changes at the beginning and the end; in (c), the two bracketed areas are largely the same but carry subtle differences.

With more precise vocabulary in music, we can describe them as: in (a), the second occurrence is the first but tonally transposed upward by a whole tone; in (b1), a crotchet in the first line is split into two quavers in the second; in (b2), the same rhythmic changes as in (b1) apply; in addition, the second line starts off-beat, and pitches of three notes are changed in comparison to (b1); in (c), the melodic motif of the first occurrence is transposed by a semitone in the second occurrence and the underlying harmony changes from B flat major to F dominant seventh, and then from F dominant seventh back to B flat major again.

Difficulties for the algorithms

The descriptions above give just one possible set of relationships between pattern occurrences amongst many. In other words, given the same musical excerpts, there might be alternative or multiple interpretations of the patterns contained within them as we showed earlier in Figure 1.1 and 1.2. From these examples, we can see the diversity in musical patterns and therefore we can begin to appreciate the substantial complexity that can come with automating musical pattern discovery. This complexity includes, but is not limited to, reconciling different time-scales, different types of patterns, and different degrees of subjectivity and ambiguity. We can only provide a limited number of examples here, but how these generalise is even more interesting: imagine all pieces of music that have been written throughout human history and all the patterns that can be extracted from them. All of these ambiguities

1 Introduction

(a) A piano étude in 4/4 time. The melody consists of eighth-note patterns. Brackets labeled "Occurrence 1" and "Occurrence 2" group the first and second half of the pattern respectively.

(b1) A folk song in common time (indicated by a "C"). The lyrics are: Daar waren drie lo - ze ge - zel - len. Brackets labeled "Occurrence 1" and "Occurrence 2" group the first and second half of the pattern respectively. The lyrics for Occurrence 2 are: die wil - den el - kan - der wat ver - tel - len.

(b2) The same folk song in common time. The lyrics are: Daar waren drie lo - ze ge - zel - len. Brackets labeled "Occurrence 3" and "Occurrence 4" group the first and second half of the pattern respectively. The lyrics for Occurrence 4 are: Die wil - den el - kan - der wat ver - tel - len.

(c) A classical piece in 2/4 time. The melody is shown in two staves. Brackets labeled "Occurrence 1" and "Occurrence 2" group the first and second half of the pattern respectively. Red circles highlight specific notes in the first occurrence that differ in the second occurrence.

Figure 1.3: Example musical patterns in (a) Piano étude (b) Folk song (3) Classical music. Brackets and bolded lines indicate pattern occurrences. Circles highlight the differences in the occurrences.

and varying interpretations pose difficulties for algorithms to work satisfactorily in all contexts.

Musical patterns: connection with repetition and variation

Musical patterns are versatile and are intuitively used in interpreting and communicating about music. One thing they all appear to have in common is that certain parts of the patterns seem to repeat. Although there are often obvious repetitions in music, the **ambiguity** in the music and **subjectivity** in human perception and cognition of patterns often obstruct a precise and all-encompassing definition of musical patterns. "Repetition sans ennui"—it is this intriguing property of music that lends

itself to the title of this dissertation. An additional problem arises when the design and evaluation of computational systems require accurate specification. Once more, we come back to see this gap between what people would recognise as a pattern and what can be computationally discovered to be patterns.

The importance of intra-pattern and inter-occurrence relations

Out of the many strategies and solutions we will be seeing in the following chapters, we would like to emphasise the importance of intra-pattern and inter-occurrence relations: looking at the relations between the pattern occurrences that make up patterns. Examining the relations between pattern occurrences can be viewed as a tool that can help to interrogate the thorny issue of capturing the definition of generic patterns, at least in part. This viewpoint also aligns well with some ideas in *Functional Programming (FP)*.

Implementation and functional programming

When diving deeper into the programming side of things, in the architecture of the actual implementation of systems that try to discover patterns, we see many aspects of patterns that one can ponder upon. Although it is clear that computational tools revolutionised many aspects of research and our everyday lives, there is usually a separation of concerns between these high-level utilities and the low-level codes, which leads to many people overlooking the relevance of software design and programming languages. We believe, however, that the alignment between the design of the implementations and their higher-level goals, and thereby how they capture the appropriate abstraction, is crucial. In this dissertation, we demonstrate this point by showing and pointing out the considerations we made in implementing our tool in a functional programming language, Haskell.

The big picture

To summarise and give a big picture of the musical pattern discovery process we will be looking at in this dissertation, we show a schematic diagram in Figure 1.4. From the agent level (algorithms and annotators), to the corpora level, to the intermediate levels (a symphony, a tune, and other intermediate levels such as a tune family and a movement of a piece), to the patterns in the piece, to the occurrences in the patterns, and eventually, to how these occurrences relate to each other, we will look into these relations between the pattern occurrences and infer back to the piece, corpora, and agent level.

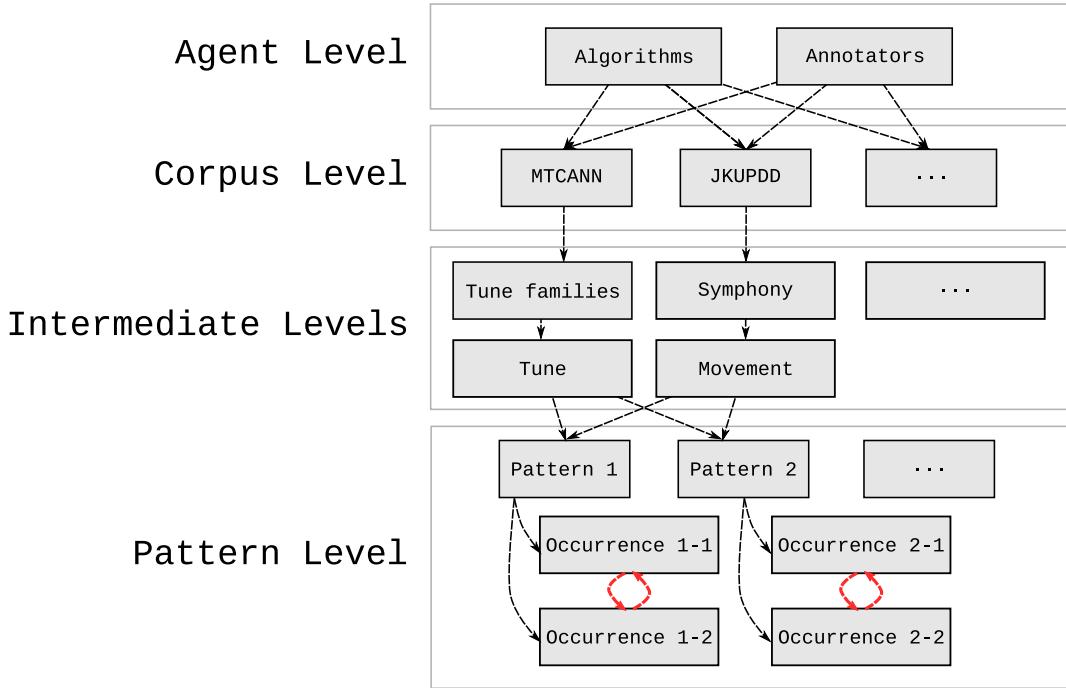


Figure 1.4: Schematic diagram of musical pattern discovery. Starting from the left, we have different pattern discovery agents that operate on different corpora, which have different intermediate levels. Within these intermediate levels, we have musical patterns and their occurrences. The red arrows represent the relations between occurrences, which will be one of the foci of this dissertation.

This chapter

In this overview, we have sketched out the background and highlighted elements of this dissertation. In the following parts of this chapter, we will detail and summarise a few important rationales and aspects of our work. We first identify the challenges . To address these challenges, we list the approaches we will be taking in this dissertation. To give context for establishing the suitability of our approach, we then give a summary of the disciplinary contexts we will encounter in the following chapters. From these diverse contexts, we distil and detail the scope of this research. Following that, we then crystallise our thesis statement and provide an overview of the structure of the dissertation. Next, we enumerate the contribution we have made. Finally, we wrap up the chapter with a note on some conventions we have embraced in this dissertation.

1.2 Challenges

With the overview of topics in mind, we can identify several challenges. Reserving more detailed background introductions for later sections and individual chapters, we sketch out six overarching challenges that we address:

- Regarding pattern, its pervasiveness and its use in different contexts is a challenge. Furthermore, the lack of annotated patterns to represent a diverse range of music poses a challenge to both designing and evaluating musical pattern discovery algorithms.
- Regarding music, the diversity of corpora and theories is a challenge. In addition, the ambiguity and lack of semantic meaning, such as those found in languages, also add to the difficulty of pinning down patterns in music.
- Regarding perception and cognition, the inherent subjectivity in how we compose, perform, listen, analyse, and teach music generates challenges.
- Regarding computation, the combinatoric explosion of possible musical events in sequences is a challenge.
- Regarding algorithms, their complexity, and as a result, if the algorithm employs black-box optimisations or other types of complex mechanisms that output hard-to-interpret results, we face the challenge of explaining and understanding them.
- Regarding multi-/cross-/trans-/inter-/non-disciplinarity, although cross-fertilisation between disciplines provides fertile soil for fostering knowledge across music, MIR, pattern recognition, and functional programming, we face the challenge of having different interests, vocabulary, and ways of thinking and reasoning between different fields.

A complicated story

These challenges intertwine with each other and form a complicated plot. To start with, people can perceive different patterns when given the same stimuli due to the inherent ambiguity of the data or the subjectivity of the recipients. For example, with a familiar music style and much exposure to the same music structures, one can realise and conceptualise the repetitions and variations as patterns (Zbikowski, 2002). In contrast, if given a complex piece of music, without any external guidance, it is possible for humans to not perceive the patterns when first hearing it, but after a few times of exposure and with external cues, one can start to perceive the underlying patterns. Finding patterns in music could be of different levels of difficulty for humans. This variance in perception contributes to the complication of how

to define "pattern", how to design and apply the algorithms and systems, evaluate and interpret the output and what annotated patterns we should gather and how to gather them. The norm in one pattern dataset could be the exception in another.

More on disciplinarity

Forming new multidisciplinary alliances is not easy. There are many conventions from one party that might not be widely known in others. For example, in the U.K., music theorists and ethnomusicologists are all musicologists, while American scholars clearly identify themselves as musicologists, music theorists, or ethnomusicologists (Cook, 2000).

Another example is the word *monad*: "In music, a monad is a single note or pitch. The Western chromatic scale, for example, is composed of twelve monads. Monads are contrasted to dyads, groups of two notes, triads, groups of three, and so on." Monads in Haskell, however, are a way to structure computations.

The last example is the very keyword "pattern". Patterns, or design patterns, in programming in general, is a family of general repeatable solutions to a commonly occurring problem in software design (Gamma, 1995). A pattern, in music, as we will dive into later, can be a group musical event. We have to disambiguate, establish, and use vocabulary in strict ways when necessary. In this dissertation, we do not make a strict distinction between music theory and musicology, and will disambiguate other vocabularies in the main text and in the glossary.

Studying these interaction spaces of different research areas, in return, bears multiple benefits, such as providing for new ideas, capturing the complexities in a larger picture and from different perspectives, and so on. This type of "multilingualism" can support the transfer of insight between fields, which is something we strive for in this dissertation.

1.3 Approaches

This project is influenced by many different areas of research, including [AI](#), [Human-Computer Interaction \(HCI\)](#), software engineering, statistics, machine learning, complex systems science, functional programming, category theory, music cognition, music theory, and musicology. The variety in the above list is not unexpected given the interdisciplinary trend in the field of [MIR](#) as well as in sciences in general. These various perspectives can contribute in their own distinct or confirmatory ways to our central topic—musical pattern discovery.

With many crucial references to musical pattern discovery, **MIR** is the basis for this research. Very much similar to this interdisciplinary area of research, we use computational methods in combination with musicology, music theories, and music perception to support our computational results. Statistical testing, **Principal Component Analysis (PCA)**, classifiers, and so on, are all common tools for understanding music data. Modelling, category theory, and software engineering perspectives from the research area of functional programming are also considered.

More concretely, relevant approaches and how they are employed are:

- Data fusion: inspired **Pattern Polling, a method we devised (PP)**.
- Pattern/data mining: previous work related to musical pattern discovery.
- Statistical analysis: used to compare between groups of patterns.
- Machine learning: classification in analysing group differences, synthetic data, with the emphasis on interpretability and explainability.
- Functional programming: using Haskell programs to model the relations between musical pattern occurrences.
- Category theory: a framework behind the implementation in Haskell.

We draw from recent advances in these areas to develop multiple kinds of analysis for musical patterns. We try to recognise the relevant discourse within various disciplines in the process.

An approach we propose towards the end of the dissertation is the use of transformations. In order to study the differences and commonalities between different annotators and algorithms, we employ a set of transformations to compare different pattern occurrences by grouping these occurrences according to their relations. The transformations we consider are musically meaningful and have been employed as compositional techniques, such as transposition in pitch and time. Using transformations, we compare the patterns from different annotators and algorithms by grouping the occurrences according to their relations, which sheds light on the potential and effective criteria the annotators and algorithms might have used to discover the patterns.

1.4 Disciplinary contexts: a summary

We can see from the preceding sections that pattern discovery has long been a part of the study of computational methods and music. In this section, we briefly summarise the disciplinary contexts on which this research is based. We only offer a high-level summary of the topics here, and each chapter may provide a more com-

prehensive overview of relevant work. Furthermore, Chapter 1 and Chapter 5 are devoted to establishing the academic backgrounds and contexts of our work.

1.4.1 Music Information Retrieval and pattern discovery

Established in 2005, the [*The Music Information Retrieval Evaluation eXchange \(MIREX\)*](#) initiative gathers tasks for comparing algorithms in [**MIR**](#). The pattern discovery task in [**MIREX**](#) started in 2016. The full name of the MIREX pattern discovery task is the "Discovery of repeated themes & sections" task.

Other typical MIREX tasks include segmentation, beat tracking, chord labelling. Evaluation metrics that have been widely used are cross-entropy, accuracy, precision, recall, F1-scores. As the patterns and algorithms we compare are closely related to this task, we will provide more information about this task in later chapters.

Methods that are commonly seen in MIR include statistical testing, [**PCA**](#), similarity metrics, and the recent rise of [**Deep learning \(DL\)**](#). We will use most of those methods in the chapters except [**DL**](#).

Outside MIR, there are many other computational methods that aim to discover, recognise, find patterns (we use all these verbs interchangeably). We will discuss these algorithms, too, as we progress to see more algorithms for pattern recognition.

1.4.2 Musicology, music theory, psychology, and cognition

One definition of music was given by Edgar Varése in the 1920s. He defined music as "organized sound". This definition separates out two aspects of music: the abstract structure (organisation) and the physical vibration (sound). This dissertation focuses more on the organisation part.

Even this concise and powerful definition is not without controversy. Speech, for example, can be seen as organised sound, but not as music in the traditional sense. Furthermore, some of the organisations employed by composers do not have to be audible by listeners, and therefore make little difference in determining the boundary of being musical. More generally, it is difficult to be exact with the meaning of the words "organised" and "sound", because they are concerned with human cognition and perception.

In the broadest terms, cognitive science is the analytical study of the human mind (Lent, 2017). Perception is mostly concerned with sensory input, but it may also be considered as a form of a cognitive process. Both perceptual and cognitive aspects

are studied under the psychology of music (Deutsch, 2019). Music is fundamentally psychological, as it can hardly be considered music if it does not involve human listening experience.

Music cognition and perception are intertwined with music theory and musicology. (Ockelford, 2017) provides a summary of the diversity of music theory and analysis:

...music theory and analysis take many different forms, and various attempts at classification have been made. Nicholas Cook, for example, divides analysis into 'traditional methods', such as those of Donald Tovey and Charles Rosen; 'Schenkerian analysis'; so-called 'psychological approaches' (principally those of Leonard Meyer and Rudolph Reti); 'formal approaches' (including the work of Allen Forte and Jean-Jacques Nattiez); and 'comparative analysis' (through Charles Adams's classification of melodic contours, for example, and Alan Lomax's 'cantometrics').

While it is possible to use specialised music-theoretic terminology for each genre and individual composer, we would like to see computational modelling that can model symbolic music more generally. From this general aspect, we are motivated by the importance of repetition and variation in music theory and psychology. In fact, repetition and variation are vitally important in music research (Huron, 2006; Margulis, 2014; Ockelford, 2017; Temperley, 2014; Volk *et al.*, 2012; Zbikowski, 2002), and they are one of the key themes of this dissertation, which is reflected in the title.

1.4.3 Functional programming, pattern, and transformation

To realise any computational idea from scratch, we need to write in a programming language. Other times, we depend on pre-existing software that we are familiar with and trust. Although writing code or using software is not always considered a scientific contributions in and of itself, it is often what we base our analysis upon.

There are many ways one can use software or implementations of computational ideas. Several advantages of using the functional programming language Haskell include the support for compositionality and modularity, ease of using higher-order functions, and a powerful type system.

Combining the thinking of functional programming and pattern discovery, we arrive at the conclusion that patterns can be examined using transformation, which can be encoded as functions and used compositionally in Haskell. Musical transformations are common in composition, analysis, and psychology, and can be manip-

ulated in Haskell with ease. We will provide more background on this in Chapter 5.

1.4.4 Broader related areas and connections

The diversity of music comes from many sources. Starting with one single concept of music, the diversity stems from one or more style(s) of a certain genre or an epoch, one or more composer(s), one or more period(s) in the composer(s)' lives, and finally end with their individual work(s). Each stage is influenced by historical events, social interactions, and our biology. An ideal algorithm would have to generalise through these aspects, which is incredibly complex and currently impossible. However, the plethora of links between the above areas is where the hidden fruits can be found.

We may also go further into the connections between the broad areas of the arts and the sciences, where most people agree that both of them are critical to human inventions. Art teaches us to see things through abstraction. Abstractions are also useful in science because they "create a new semantic level in which one can be absolutely precise" (Edsger W Dijkstra). Although science is more problem-driven, analytic, and objective, and arts are more artefact-driven, generative, and subjective (Pease *et al.*, 2019), they have the common pursuit of intuition, imagining something that does not yet exist and potentially incomprehensible to others (Dibbets, 2002). It is little surprise that there have been numerous connections made between music and mathematics, mathematics and computation, and were often taught together in the past.

Based on mathematics and computation, machine learning and AI contribute the most when it comes to pattern recognition. Conversely, the diversity of music provides opportunities to test the algorithm for true generalisability. We would like to elicit convergence and create synergy between these different strands of research by approaching our topic from various perspectives. Several related concepts, for example, occur in various fields and are referred to by different names. Such concepts tend to be used over and over. For instance, a bottom-up approach can be more closely related to system 1 thinking (Kahneman, 2011), semantics, surface structures, and melodic features, while a top-down approach has more affinity with system 2 thinking (Kahneman, 2011), syntax, abstractions, and chords. These connections are also linked to the recent progress in incorporating logic into probabilistic and neural approaches. By marrying symbolic and sub-symbolic methods into an integrative or hybrid paradigm, it offers an opportunity for the algorithms to escape the robustness issues they currently suffer from.

1.5 Scope

In this section, we explain the concrete scope of the dissertation. Our analyses are scoped by using monophonic, (semi)symbolic MIDI data, MIREX algorithms and format, and have a focus on intra-pattern and inter-occurrence analysis. We believe that within this well-defined scope, albeit small in contrast to the complexities of such large topics as patterns and music, we can make direct and deep contributions to the field. We expand the discussion on our choice of scope below.

1.5.1 Why monophonic

We concentrate on and use monophonic data in this dissertation for three reasons. First, monophonic music removes the complexity of polyphony, paraphony, or heterophony. Pattern discovery in monophonic music is not a trivial problem either, as there are many intriguing melodic designs in folk music, jazz, pop, classical, and plenty of research focuses solely on monophonic music. We use monophonic music as the first step toward greater complexity.

These genres lead us to our next point, which is the fact that sometimes the monophonic aspects of music can be orthogonal to its harmonic aspects, such as in many types of world music and atonal music. In the situations where we see both the harmonic and melodic aspects, it is also possible to infer the characteristics of harmonic ones from the melodic surface.

Third, monophonic music is simple to create and arguably the origin of all music. As humans, without any instruments, singing is the very first way we can make music by ourselves, monophonically. Monophonic music is also more parallel to speech and language than non-monophonic music.

However, we do bear in mind that we will miss the cross-voice patterns, harmonic patterns, and abstractions to chords. This underlying harmony is often a more efficient representation, and using monophonic data might be extra challenging in some cases if we do not consider the underlying harmony.

1.5.2 Why symbolic

We focus on (semi)symbolic data because it provides us with cleaner information than acoustic signals. Converting from acoustic information to symbolic representation is known as automatic transcription, which is a research topic of its own.

More specifically, we chose to use *Musical Instrument Digital Interface (MIDI)* as the main input format, because of its ease of playback and the fact that it is the stan-

1 Introduction

dard for a lot of software. MIDI is sometimes also known as (semi)symbolic rather than symbolic because it lacks the power to express certain constructs in sheet music, such as pitch spelling. With certain algorithms, we use other types of input, the MIREX Lisp and CSV formats, which added in morphetic pitch (Meredith, 2006).

1.5.3 Datasets and generalisability

The size of datasets matters as it is related to the problem of overfitting, which we will explore more in Chapters 3 and 4. With the recent investigation into the biases in music theory (Ewell, 2020), there has been an outcry that we must confront continued reliance on small, unrepresentative corpora. In music theory and musicology books, some typical numbers of pieces and examples analysed are approximately 40 pieces for Schenker, 288 examples for Caplin, and 552 examples for Hepokoski Darcy (London, 2021). Amongst these, 25/50, 288/288, and 383/552 are from the composers Bach, Haydn, Mozart, and Beethoven (London, 2021). It also was argued that a lack of diversity could have significant effects on how we think about musical structure (how we "do music theory") and on how we think about how we hear and understand music (how we "do music cognition"). We, therefore, strive to include a diverse range of music data, but some of the available datasets we use do have a bias towards these classical Western composers as well.

An appendix is included at the end of this dissertation containing details on the datasets used in the various chapters. We also provide a summary of the dataset characteristics when used in each chapter.

1.5.4 Format of patterns

The main format we use to encode patterns and their occurrences is the MIREX format. Algorithms submitted to the MIREX platform all produce this type of encoding as shown in the example after this paragraph of text. The format always starts with `pattern1` on the first line, `occurrence1` on the second line, followed by the timestamp and MIDI pitch value of the notes consist of the patterns on the following lines. Each note takes a line, until the end of the pattern occurrences, where the next heading of `pattern-i` or `occurrence1-i` occupies the next line. The encoding we use are widely and currently used in the research of pattern discovery in music (de Reuse & Fujinaga, 2019; Meredith, 2019).

```
pattern1
occurrence1
17.00000, 74.00000
```

```

17.50000, 74.00000
18.00000, 74.00000
21.50000, 74.00000
22.00000, 72.00000
occurrence2
26.00000, 69.00000
26.50000, 69.00000
27.00000, 69.00000
30.50000, 69.00000
31.00000, 67.00000
occurrence3
46.00000, 71.00000
46.50000, 71.00000
47.00000, 71.00000
50.50000, 71.00000
51.00000, 69.00000
occurrence4
...
occurrence19
pattern2
occurrence1
...
pattern45
...

```

1.5.5 Inter- Intra- opus, patterns, and occurrences

According to (Conklin, 2010), there are two forms of pattern discovery: inter-opus (discovering patterns recurring across a number of pieces in a corpus) and intra-opus (discovering patterns repeating in a single piece). To simulate intra-opus from inter-opus, one can concatenate a number of pieces into a single file. However, this method does not make them exactly equivalent, as patterns may be discovered at the boundaries of concatenations. In addition, the order of the concatenation will become important, and the memory requirements for the two tasks will be significantly different.

The human-annotated patterns in [The Meertens Tune Collections: The Annotated Corpus \(MTC-ANN\)](#) are inter-opus, while other annotations are intra-opus, so we

1 *Introduction*

do have a focus on intra-opus pattern discovery. We also use the simulation method described in the previous paragraph.

This same distinction between inter- and intra- can be applied to defining pattern, too. An inter-pattern type of definition focuses on how to define and distinguish one pattern from another, and an intra-pattern type of definition focuses on the relations between occurrences inside a single pattern. Our transformation-based approach is very much suited to the intra-pattern definition.

We can follow this thought process further to pattern occurrences. When we compare between occurrences using transformations, we are making an inter-occurrence comparison. While we do not do so in this dissertation, if we dive into the structures carried by the occurrences themselves, it would be called intra-occurrence comparison. This type of intra-occurrence comparison can also be interesting because when a single occurrence is invariant under a transformation, it could mean that there are substructures inside the occurrence. We will leave this to future work and focus on inter-occurrence comparisons in this dissertation.

1.6 Thesis statement

Now that we have defined the scope, inspected disciplinary contexts, drafted the approaches for the anticipated challenges, and declared our motivations, we can summarise a thesis statement for this dissertation. In this dissertation, we strive to make the argument that

Patterns, as a type of highly subjective and ubiquitous abstraction, have important connections to transformations, which can be explored compositionally using a functional programming language to reveal implications for designing musical pattern discovery algorithms.

We unpack this sentence into three main aspects—understanding patterns, making connections to transformations, and using a functional programming language. Chapters 2-4 focus on the understanding of patterns and how humans and algorithms discover them. Chapter 5 makes the connection to transformations, and Chapter 6 implements this idea in a functional programming language, Haskell. Chapter 7 examines the analyses that can be performed using our implementation and suggests further improvements. A more extensive description of the outline of this dissertation is provided in the next section.

1.7 Dissertation outline

The overall structure of this dissertation takes the form of eight chapters. Figure 1.5 shows this structure and the connections between the chapters. In the following subsections, we will go through the chapters, stating their connections to our publications (also see Appendix E) and highlighting their contents and aims.

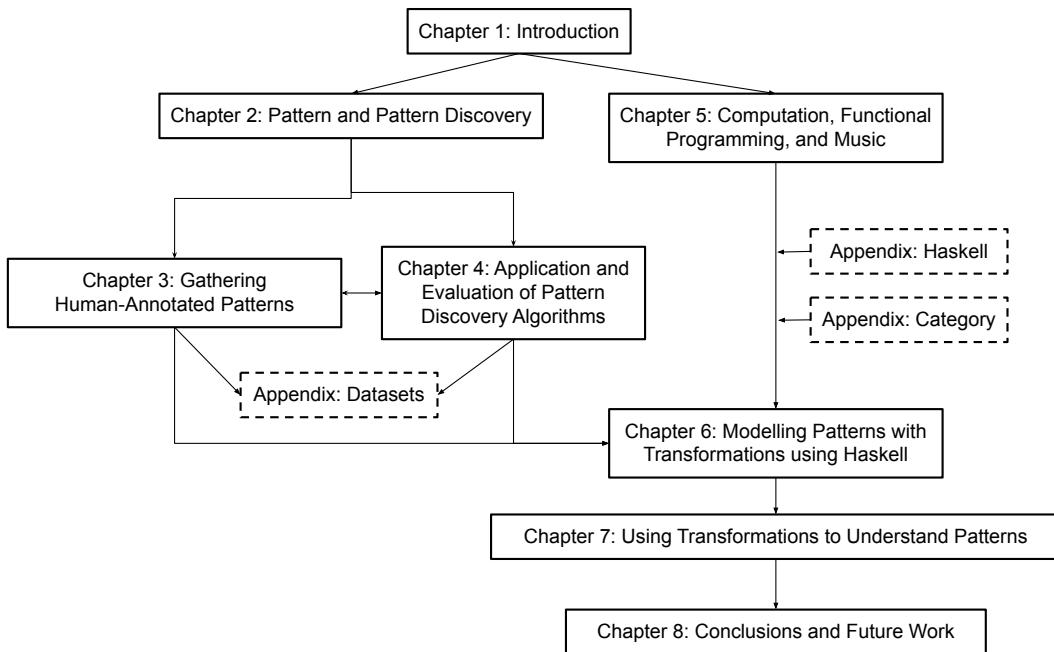


Figure 1.5: Thesis structure—the relationships between the chapters in this dissertation.

Introduction

This chapter gives the reader a bird's eye view of the dissertation. We begin by describing the motivations, challenges, and approaches used in this dissertation. Then we go over the research landscape and the scope we take. This chapter also puts forward our thesis statement and the outline of the dissertation to substantiate the thesis. At the end of this chapter, we summarise our contribution and give the conventions we use for this dissertation.

Pattern and Pattern Discovery

This is a chapter where we introduce relevant work centring on the concept of pattern. Starting with the broad usage of the word "pattern", we include examples and discuss this concept drawing from the perspectives of several different sources. With some thought experiments, we tap into the factors that play a role in human percep-

tion and cognition of patterns. Finally, we create a list of related concepts to establish their (ir)relevance to this dissertation.

Gathering Human-Annotated Musical Patterns

This chapter is based on the following published papers and master thesis: (Ren, Koops, *et al.*, 2018; Wells, 2019), and ...

In this chapter, we discuss a number of issues regarding human-annotated musical patterns. We also release digitised pattern annotation data from three experiments in the **JSON Annotated Music Specification (JAMS)** format for facilitating future research. Two annotation tools and three experiments are compared and discussed with respect to their functionality and the datasets produced.

Application and Evaluation Pattern Discovery Algorithms

This chapter is based on the following published papers: (Ren *et al.*, 2020; Ren *et al.*, 2017; Ren, Volk, *et al.*, 2018).

We examine a number of musical pattern discovery algorithms. To compare and evaluate them, we propose five methods:

- Visualisation: we devise new visualisation methods as well as using existing ones to examine a large number of patterns.
- **PP**: We devise PP to integrate the output from various algorithms, following fusion and ensemble methods commonly used in improving pattern recognition accuracy. We demonstrate that there are encouraging correspondences between the combined results and human annotations, but that discrepancies between the output from different algorithms pose a major barrier to any significant improvement.
- **Comparative Classification, a method we devised (CC)**: To verify the discrepancies between the algorithms, we leverage the discriminative power of classification algorithms to compare features extracted from algorithmically discovered, hand-annotated, and randomly selected patterns. We discover that rhythmic features play the most important role in differentiating between these pattern groups.
- Synthetic data with planted patterns: To better understand the behaviour of various algorithms, we examine them using synthetic data, which is random data with musical patterns and transformations planted within it. We discover that some algorithms perform as expected while others do not.

- Prediction: we set up a prediction task for MIREX, in the hope of examining patterns for a concrete application.

Computation, Functional Programming, and Music

This is a connecting chapter focusing on the intersection of computation (especially functional programming) and music. We introduce Haskell’s connections with music. With the consideration of analysing musical patterns, we make a connection to transformations. We examine a variety of studies to support the suitability of modelling patterns using transformations.

Modelling Patterns with Transformations using Haskell

This chapter is based on previously published work of (Melkonian *et al.*, 2019). Using transformations, we describe a framework for comparing musical patterns, utilising well-known abstractions from category theory. We also provide a Haskell implementation of the model—Pattrans—in the form of an embedded *Domain Specific Language (DSL)*.

Using Transformations to Understand Patterns

This chapter is partially based on the published work (Melkonian *et al.*, 2019), and contains results we are preparing to submit. We showcase a minimal query language for querying patterns up to transformations in Pattrans. Using the data obtained by using Pattrans, we explore how to use musical transformations to relate and classify musical pattern occurrences. We also describe some extensions that can be made to Pattrans.

Conclusions and Future Work

This chapter will summarise the dissertation and discuss a few potential applications as well as future work.

1.8 Contribution

To summarise the chapter-wise contributions in the previous section, we categorise them into two main types of contribution:

- Creating tools and datasets.

1 *Introduction*

- Analysing existing musical pattern discovery algorithms in order to address questions about musical patterns computationally.

More specifically for datasets and tools, we create three human-annotated datasets of musical patterns, and make a Haskell package for checking patterns. For analysis, we propose novel comparisons between algorithmically extracted and human-annotated patterns using *State-of-the-Art (SotA)* methods (PP, CC, synthetic data, feature analysis). We compute the transformations, features, and metrics for musical patterns and visualise them in various ways. A variety of findings and observations are made, revealing a wide range of (dis)similarities between algorithmically extracted and human-annotated patterns, which paves the way for a wider range of discussions and applications.

In addition, we make other lighter contributions, including:

- Serving evaluation tasks on the MIREX platform: we served as task captains on two MIREX tasks involving musical patterns.
- Starting interdisciplinary discussion: we discuss topics that span a number of disciplines. More specifically, for example, we model musical repetition and variation with functional programming to initiate the dialogue between the implementation and the analysis sides of research.
- Examining a wide range of literature regarding pattern and transformation: we consider a variety of literature to synthesise and support our definitions and approaches to studying musical patterns.

1.9 Notes on reading this dissertation

We have eight chapters in this dissertation. Each chapter begins with a few quotations that are important to the chapter's theme. We then lay out the structure of the chapter and relations with other chapters before the overview section begins. The overview sections will then paint a broad context for more detailed discussions, relevance to previous work, experiments (if any), and findings. Each chapter finishes with a summary that briefly reviews the conclusion and contribution of each chapter. Finally, we open up discussions on more tangential but also important topics in the chapter. We try to keep the chapters as modular as possible and aim to convey the intuition behind complex ideas.

In addition to the main chapters, we have a preface, other supporting appendices, and backmatter. An acronym list is provided at the beginning of this dissertation to provide an overview of the expected relevant keywords and newly coined ones. The

glossary at the end of the dissertation is made for explaining concepts that appear in the dissertation.

Throughout the remainder of this section, we provide a short description of terms used in the rest of this dissertation for fluidity of reading. More extensive and precise meanings of the words will be given in the main text and glossary. We have already encountered the majority of these words where we used their meaning loosely in this chapter. Here, we summarise what we mean when we say:

- Musical pattern: regularities in music. (We will spend the second chapter discussing this idea to give a more comprehensive view and a working definition for this dissertation.)
- Pattern occurrence: an excerpt of music that is representative of a pattern.
- Musical transformation: how an excerpt of music transforms into another excerpt of music.
- Annotation: an excerpt of music that has been annotated by a person. In other literature, it may refer to both algorithmic annotations (what we term algorithmically discovered patterns) and human annotations.
- Inexact and exact repetition: we treat inexact repetition as variations and not repetition (please see more discussion in Section 2.3.2)

To describe musical events, we use the following words frequently:

- Pitch: the frequency of a sound, usually discretised using MIDI numbers
- Interval: the differences between two pitches
- Duration: how long a pitch lasts
- Note: a combination of pitch and duration
- Dynamics: loudness

To reason about how people perceive music, we use the following concepts:

- Ambiguity: more than one valid interpretation is possible given the same data (intrinsic to the data)
- Subjectivity: interpreting the same data in multiple ways (intrinsic to the observer).

To encode and compute in the context of music, we use:

- Feature: an aspect of consideration. A feature vector is vector of numerical features.
- Representation: an assemblage of features
- Data: a set of numbers or objects that relate to a problem
- Prior: the initial belief that a variable follows a certain probabilistic distribution

In [FP](#), the following concepts are important:

1 *Introduction*

- Data type: a variable with a set of constraints
- Side effects: altered non-local states, caused by non-pure functions

Some terminology we do not make a strict distinction between includes:

- **DSL** and package: mostly related to Chapter 7 and 8, we use them interchangeably.
- Pattern and occurrence: patterns have pattern occurrences. Although we make an effort to distinguish the abstract idea of pattern, which is a name for a group of occurrences, it is sometimes self-evident when we refer to an occurrence as a pattern.

Finally, we use "a type/kind of pattern" without referring to the specific concepts and definitions in Haskell, but in future work, it may be useful to make type-level modelling of patterns. In terms of spelling, we will use British conventions in our main texts, and use any other conventions that come in quotations unchanged.

Chapter 2 Pattern and Pattern Discovery

Humans are pattern-seeking story-telling animals, and we are quite adept at telling stories about patterns, whether they exist or not.

– Michael Shermer

In this chapter, to further motivate the analysis of musical pattern, we examine the concept of *pattern* from a variety of perspectives. By synthesising and differentiating the various usages of this concept, this chapter provides the background of this dissertation using diverse viewpoints. We start with a broad overview of the topics we cover in this chapter. Then we examine a range of definitions of patterns, from the most general to the very specific. Following these ideas, to further highlight the importance and the complexity of the concept of pattern, we show the relations between this concept to other key concepts in [ML](#) and [MIR](#). We then place this concept in the musical context of music theory and musicology, where relations between the concept of pattern and other terminologies are examined. The aim of this chapter is to dissect the meaning of pattern and arrive at a working definition of musical pattern for this dissertation. We conclude the journey of this chapter by giving our working definitions, a summary of the chapter, and further discussions into other tangential topics. As this chapter presents material from many sources and contains thought experiments designed to further our understanding of patterns, we recommend that readers who are less interested in such exercises read it briefly and return to the details later if so desired while reading other chapters.

2.1 Overview

"Pattern" is a broad term. It is used in philosophy, engineering, science, the arts, and a variety of other domains. Everybody has their own personal and often hidden conception of patterns.

2 Pattern and Pattern Discovery

In mathematics, we see patterns in tessellation, visual motifs, symmetry, and fractals. In programming, there are similar constructs such as pattern matching, design patterns, and subexpressions. In nature, we have the process of pattern formation, and we see patterns in trees, dunes, and ocean waves. The Dunning-Kruger effect (Dunning, 2011) of this word seems to be strong: at first, it seems to be universal, simple to understand, but when we study and discuss it further, more and more diverse, nuanced, and complex points arise.

Why is the concept of pattern important?

"The ability to perceive and manipulate patterns is a fundamental part of human intelligence in general, and of scientific discovery in particular."
(Meredith, 1991)

In addition to claims such as this, one reason that pattern is important is that more and more data is becoming available. Pattern recognition algorithms have benefited from this fact and made one breakthrough after another on many subjects, such as computer vision and bioinformatics (Bishop, 2006). One definition of pattern recognition is

"...the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories" (Bishop, 2006)

This seems to equate a pattern to
regularities that can be used to take subsequent actions.

However, this definition is more descriptive than functional: it does not help us to find them. What other definitions are out there?

Data and abstraction, repetition and variation, and how we find them

Let us take a step back and consider the meaning of data again. From data, we obtain information, knowledge, and wisdom through abstracting and connecting them from one level to the next. Here, the word "abstraction" means erasing irrelevant detail and organising relevant building blocks. For instance, when we draw a circle to represent a ball, we abstract out the most significant aspects of a ball to a basic drawing; when we use the colour red, we abstract from a band of optical frequencies; when we draw notes on a stave, we abstract out the pitch and the duration in music. These abstractions are fundamental to understanding and having a strong connection to what we often refer to as a pattern.

Another less intimidating way to look at concepts such as regularity, abstraction, and pattern is in terms of repetition and variation. It is common to see patterns repeat and vary in many contexts, notably in floor tiling, minimalist music, and zebra stripes. Conversely, if something repeats or varies in a certain way, the repetition or the specific variation may be construed as a form of regularity, or abstracted away, or form a pattern. In this light, patterns appear more intuitive.

One might believe that repetition and variation are too simple, common, dichotomous, and tautological for any serious investigations. We will discuss why, in the next two paragraphs, using repetition and variation as a starting point for thinking about patterns could harbour more richness and complexity than one might think.

Repetition and variation are common concepts that are polar opposites of each other. Rather than a dichotomy, we hold that they can be flexible concepts, and a spectrum of connections exist between them. They are flexible because, although there is a platonic ideal of exact repetitions, they rarely occur: even the same piece of music, performed in different countries, concert halls, and with different instrumentalists, can sound different to different listeners. Given these inexact repetitions, an array of variations naturally arise. Ultimately, repetitions and variations are two sides of the same coin, and therefore tautological, yet the coin is *transparent* and the distinctions between them are somewhat *softer*. The *transparency* stems from the fact that an inexact repetition can be viewed as a variation, and a minor variation can be viewed as repetition, but the emphases are different. The *soft* boundaries stem from cases where spurious repetition might preserve just enough to make it look like repetition to some while it looks like variation to others, which makes a hard boundary almost impossible to find. We summarise these relations in Figure 2.1, which also summarises what they mean for this dissertation to prevent further confusion: if one of the attributes of the data is changed, it is a variation; similarly, if there is a *slight* change in one of the attributes of the data, no matter how small, it is also a variation. With these relations and clarifications, we may explain patterns with varying degrees and types of repetitions and variations.

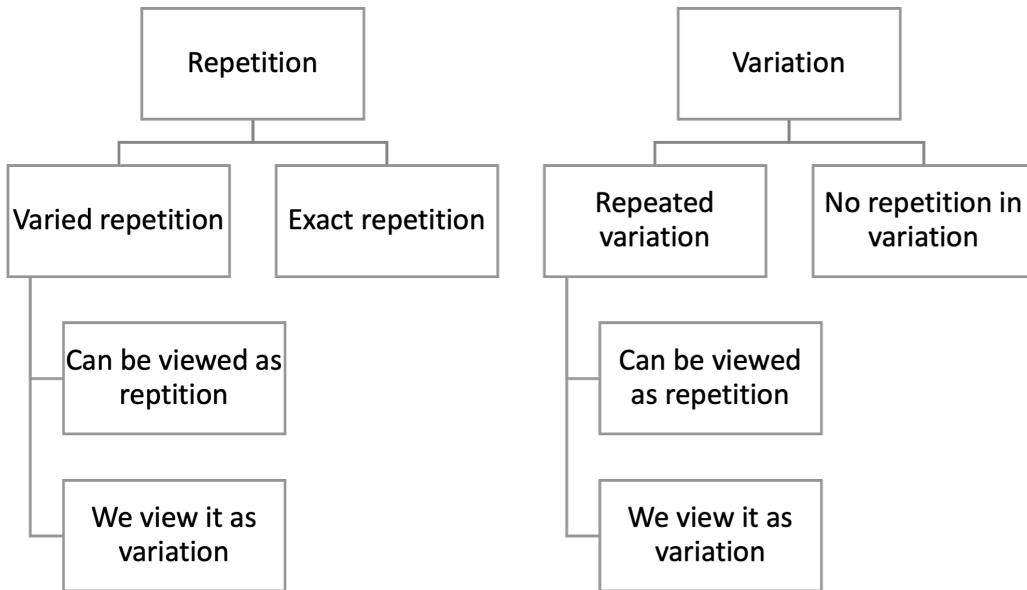


Figure 2.1: A diagram of how variation and repetition can be entangled and our stances. At the second level, we only show the two extremes: there are spectra of degree in-between them.

Once we make the connection that repetition and variation are useful starting points for investigating pattern, it leads us to a tie with our perception and cognition. The historical, evolutionary, and biological origins of perceiving and cognising patterns have been widely commented on. Benoit Mandelbrot once said that "people want to see patterns in the world. It is how we evolved." (Mandelbrot & Hudson, 2007). Simeon Lockhart Nelson wrote that "seeing patterns is really important for survival and for full engagement with the world; it is a way of understanding complex and confounding phenomena." (Lockhart Nelson, 2017). It is rare to find any exact repetitions anywhere, and yet, we can express and communicate while having a shared understanding as to which part of the repetition to pay attention to and which part of the variations to disregard. In more literary words, patterns were described as being in the middle between subjectivity and objectivity:

".. (pattern is) an emergent epiphenomenon arising out of the interaction between subject and object. Its meaning needs to be brought to light, to be interpreted or experienced." (Lockhart Nelson, 2017).

Musical patterns

The discussion so far has dealt with patterns in general; let us now narrow our focus to patterns in music. When we zoom in on music, we can observe some fascinating characteristics.

When speaking of musical patterns, a diverse range of examples might come to mind, such as the purposeful patterns in études, the slick patterns in jazz, the elegant patterns in minimalist music, the textbook examples of fugues, a few notes from folk music, and themes and motifs from classical music. If one has a stronger affinity for music theory, constructs such as schemata, leitmotif, subject, and counter-subject might come to mind.

These musical patterns can be the inventions of scholars or cultures. Through immersion, engagement, and self-directed practice and interaction with music, and even through something completely unexpected and unrelated to music, this process of pattern finding can be influenced.

Musical patterns have been described in different ways, too. Assuming motifs are a type of pattern, the following are several different descriptions of a motif. According to Webern, they are "the smallest independent particle in a musical idea" (Webern, 1963). For Whitehead, they are a "structural unit possessing thematic identity" (Whitehead & Price, 2001). Schoenberg commented from a different point of view that it is "a unit which contains one or more features of interval and rhythm [whose] presence is maintained in constant use throughout a piece" (Cambell, 2010). They used an array of words such as structure, theme, segment, unit, and many more to describe this one type of pattern. What would we use to connect them together?

As before, we make the connection that all kinds of patterns share a certain degree of repetition and variation. This time around, the focus is on music. Given the importance and relevance of repetition and variation in music, in this dissertation, we take the studies behind musical repetitions as one of the inspirations for this research, (Huron, 2006; Margulis, 2014; Ockelford, 2017; Temperley, 2014; Volk *et al.*, 2012; Zbikowski, 2002). We do not, however, intend to be as comprehensive as the literature listed above about this topic of repetition and variation, as several books in the above list capture them in great depth.

Patterns and their repetition and variation in music

Repetition and variation in music are much hazier than they are in the visual and language domains, because in the auditory domain, everything is fleeting and there are rarely fixed semantic meanings attached to pitches and durations in music. While research has been conducted on the perception and cognition of musical repetitions and variations, not all concerns have been addressed, particularly in terms of patterns.

There are two approaches to getting a deeper understanding of the patterns in existing music: one can either ask listeners to point them out to us (the data-driven

2 Pattern and Pattern Discovery

bottom-up approach), or one can theorise and define *a priori* the interesting patterns to find (the top-down rule-based approach). For the first approach, one needs to conduct general listeners' annotation experiments, which will be the topic of the next chapter. For the second approach, one needs to examine different theories of music in further detail, which is what this chapter and Chapter 5 would do.

Musicians have been found to agree to a degree on the importance of patterns in music, although this agreement is not always perfect (Collins, 2011). At times, musical experts can be completely polarised (Giraud *et al.*, 2016; Ren, Koops, *et al.*, 2018). Several contentious questions include the following:

- When does a pattern begin and end?
- How important must a passage be in order for it to be called a pattern?
- What is the minimum length of a pattern?
- Have all of the occurrences of a pattern been discovered?

These disagreements manifest themselves in listening experiments. The conceptual boundaries that people mentally draw while listening to music are important, and they are often contentious but meaningful. As with drawing boundaries on a string of ones and zeros yields all that we see on our computers, drawing different boundaries in music results in different interpretations, which allows the listening experience to be subjective. We will discuss later in the chapter how patterns connect to other concepts such as boundaries, as well as several cognitive dimensions of how we find and conceive patterns.

Leaving the passive, listening-centric topics aside, let us consider a more interactive music experience than merely listening to music that already exists. Depending on how one interacts with music, the control of repetition and variation varies too. When we listen to music, we have no power over repetition. The listening effort is mainly spent on recognising them in order to grasp an understanding of the structure of the piece. When creating music, however, repetition may be seen as a strategy, a playful experiment, an intentional emphasis, or an unintentional coincidence. When improvising music, repetition and repetition-based variations may be used to express affirmation and consensus. Although the emphasis of this dissertation is not on music composition, we will consider certain perspectives of music-making and make appropriate connections to our findings regarding musical patterns.

Connection to algorithms

All these aspects above are something to keep in mind when automating the pattern discovery process. If we would like to create a generalisable algorithm to discover

and reuse the patterns in our data, we will run into the question of how to bridge abstract perception, reasoning, and actions of humans and machines. We will make some links to algorithms in this chapter, but will leave Chapter 4 as the main chapter to discuss algorithms.

This Chapter

For the rest of this chapter, we first examine an array of definitions and usages of the word "pattern". As patterns may be related to concepts such as repetition, variation, features, similarity, and hierarchy, there is a need to connect the concept of pattern to these similar concepts to disentangle them, which we will do towards the end of this chapter. Combining what we examine in this chapter, we put forward our working definition of a pattern for this dissertation. This will be followed by a summary and a discussion section at the end.

2.2 Pattern as a concept

The concept of pattern is both all-encompassing and enigmatic. We see it in many disciplines, and it relates to other important concepts such as abstraction, repetition, and variation. For a number of reasons, it is a perplexing term. One reason is that it is often used to refer to many things simultaneously: occurrences of patterns, patterns of patterns, patterns of patterns of patterns, are all sometimes abbreviated to simply "pattern". Another reason is that people do not express clearly which assumptions they are making and depend on implicit definitions instead. There might also be cases where no one can give a definition *a priori*: only when someone points out the existence of a pattern, it becomes simpler to pick out, but we simply do not know where it is prior to this realisation. As we discussed in the last section, patterns seem to possess both subjectivity and objectivity—"Pattern is both a function of our perception and an attribute of the world.", as put in (Lockhart Nelson, 2017).

This section is dedicated to providing multiple viewpoints on the concept of pattern. In our context, having these concepts of what a pattern is can be important in terms of what it might entail for the design of the annotation experiments, algorithms, and their evaluation. With these different angles in mind, we will attempt to summarise our definition of a pattern at the end of this chapter for these purposes.

From daily sources

Colloquially, the word pattern is used in a range of ways. There are patterns of colours, patterns of behaviours, patterns of sound, emerging patterns, destructive patterns, just to name a few. Many types of pattern exist in nature and the in designs that surround us, including spirals, meanders, dunes, bubbles, tessellations, and more.

The *Oxford English Dictionary (OED)* gives the word pattern more than ten meanings and use it to define over a thousand other words. Although our aim is to be broad, it would not be efficient to list all definitions. The most relevant two definitions out of the ten are "A regular and intelligible form or sequence discernible in the way in which something happens or is done." and "An example for others to follow." (Oxford, 2021). This seems to be in accordance with what we discussed in the last section in terms of repetition and variation. Let us bear this in mind and consider the context of music and relevant research next.

Various usage in writing about music

Musical patterns are used to summarise and describe music. In addition to what we discussed in the overview, below are a few other uses of the word pattern in the context of music:

"Patterns are to music what words are to language: a vocabulary. How can you speak a language without a vocabulary?" (Grout, 2020)

"Patterns are one of the most fundamental ingredients of music and so students with a solid pattern vocabulary will be much better as both readers and players." (Topham, 2020)

"A melodic pattern ... may be formed by any group of notes that has melodic plausibility." (Slonimsky, 1999)

Musical patterns are formed with a special group of notes. A connection is made with language, and therefore, communication. In fact, complexly patterned rhythms alone can be used as a language for communication (Clarke, 1934).

Once we verbalise a concept, it becomes a reusable mechanism and an independent piece for perception, reasoning, or action. Patterns in music, in this sense, could be something similar—a chunk of music that can be used to predict, to compress, and to understand.

One thing clear is that "pattern" is a "suitcase word"—there are lots of meanings packed into this one word. There is a variety of work that uses computational ap-

proaches with a particular focus on the term "pattern", resulting in different definitions, as we can see next.

Definition from previous works

Zooming in to the field of MIR, we can see different definitions such as:

- "A melodic pattern is defined by a set of either identical or 'equipollent' (i.e., significantly similar) sequence segments." (Rolland, 1999)
- "The term 'pattern' here basically refers to N-grams, i.e., subsequences, of melodic abstractions." (Pfleiderer *et al.*, 2019)
- A mathematical definition is given by (Collins, 2011) and ("2014:Discovery of Repeated Themes & Sections - MIREX Wiki", 2014): "a pattern is defined as a set of ontime-pitch pairs that occurs at least twice (i.e., is repeated at least once) in a piece of music", which is also the definition used in the [MIREX](#) task.

At the very least, these definitions seem to agree that a pattern is made up of a few notes that repeat exactly or inexactly. This commonality leads to two issues, at least. The first is to determine when an inexact repetition ceases to be a repetition. The second is to distinguish patterns that are trivial or created by chance from patterns that are more worthwhile and created by design.

Both issues have also been discussed in previous works of musical pattern discovery algorithms. Regarding the distinction exact and inexact repetition, we see in (Forth & Wiggins, 2009) that

... repetition may exist in many forms beyond the exact repetition of musical events in sequence. For example, melodies may still be perceived as instances of the same basic melodic motif despite being transposed in pitch or transformed in time. Indeed, perceptual similarity may pertain for any individual listener under an arbitrary number of processes of elaboration and transformation. In the context of computational analysis, therefore, careful consideration must be given to the notion of pattern equality.

Regarding the difficulty to find more meaningful repetition, it was mentioned in (Meredith *et al.*, 2002) that

Many music analysts and music psychologists (see, for example, Bent & Drabkin, 1987; Lerdahl & Jackendoff, 1983; Nattiez, 1975; Ruwet, 1972; Schenker, 1954) have stressed that the identification of perceptually significant repetitions is an essential step in the process by which an expert listener interprets a musical work. ... However, the vast majority of exact repetitions in music are not perceptually significant (see section 8 below).

Therefore, the task of developing an algorithm that isolates perceptually significant repetitions in music involves formally characterising what it is about these interesting repetitions that distinguishes them from the many repetitions that the listener does not notice and the analyst does not consider to be important.

We agree with the above views and note that a short sequence, say of two successive notes differing by a whole step, is likely to occur often in all kinds of music by chance, and we would naturally doubt whether this is a meaningful pattern where a composer thought through it as a distinctive figure worthy of repetition and development. Longer note sequences that occur repeatedly are more likely to be a product of design than of chance, but they are more likely to have complex changes between occurrences that render the inexact repetition cease to be a repetition. Throughout the chapters, we have these two issues as a central theme of our discussion.

Various usage in other research

Definitions from other disciplines of research may come in helpful to situate and navigate the above definitions, too. In visual analytics, a pattern has been defined as "a combination of multiple interrelated elements of two or more data components that can be represented and treated as a unified whole." (Andrienko *et al.*, 2021) In the information systems field, "A pattern is a generalizable reusable solution to a design problem" (Besheli, 2018). In psychology and life sciences, it has been said that "a pattern is a simpler representation of something" (Robertson & Combs, 1999). In pattern-oriented modelling of complex systems analysis, patterns are defined to be "characteristic, clearly identifiable structures in nature itself or in the data extracted from nature." (Grimm *et al.*, 1996). In mathematics, pattern theory was proposed, and the goal was to describe knowledge of the world as patterns using mathematical formalism. (Mumford, 1994) define pattern theory as: "the analysis of the patterns generated by the world in any modality, with all their naturally occurring complexity and ambiguity, with the goal of reconstructing the processes, objects and events that produced them."

From these definitions, we can see various foci due to the diversity of the disciplines. We do recognise some of the topics we have already discussed, such as repetition and regularity. We will see later in this chapter and in subsequent chapters that ambiguity and complexity will also be discussed.

Intensional and extensional definitions, necessary and sufficient conditions

Having reviewed a series of definitions, let us distinguish two common approaches to giving definitions. Intensional definitions are given by necessary and sufficient conditions. Extensional definitions are given by a compilation of things that may come under the definition. So far, the majority of what we have seen is intensional. An extensional definition stems from a dataset, something like the figure examples given in Chapter 1, in affinity to the data-driven approach.

These two types of definitions are also referred to as intensional specifications and extensional realisations. We introduce them here not to have a full ontological and epistemological debate, but to acknowledge that *pattern* can be defined in at least these two different ways.

They have a link to two common approaches in the ML—top-down and bottom-up approaches (Meyer-Vitali *et al.*, 2019). Top-down approaches are commonly known as approaches starting from first principles, such as rule-based models and symbolic AI. Bottom-up approaches are based on examples—the individual data points. Bottom-up and top-down approaches can also be compared with the type-I (naturally defined) and type-II (formally defined) categorisation process (Zbikowski, 2002). There is a growing amount of research being done to link these two types of methods (Meyer-Vitali *et al.*, 2019).

While learning algorithms are based on extensional definitions and have seen great success, we put more focus on intensional definitions for this dissertation because they give more room to reason about patterns. Moreover, a list of extensional entities is arguably less efficient than the intensional properties that could summarise them. We do believe that a mixture from both sides would be ideal so that both the intensional and extensional definitions could come together and verify each other. We will indeed examine some machine learning algorithms that are data-driven and construct a more intensionally defined system to examine them.

For intensional definitions, the next question to ask is what are the typical variables in the necessary and sufficient conditions for a passage in the music to be identified as a pattern. Sufficient conditions provide the predicates of the purposes of patterns, showing possibly some rules and algorithms to extract the patterns. Necessary conditions are characteristics, more likely to be inferred from statistical data analysis. Necessary conditions provide clues as to what is more likely to be musical patterns, and they are also filters as to what is preferred not to be regarded as patterns in a certain type of corpus or for a specific purpose. For example, there have been filters based on length, frequency, spacing, and similarity (Janssen, 2018). When a condi-

2 Pattern and Pattern Discovery

tion is both sufficient and necessary, we arrive at an "if and only if" condition, which we have not come across in the literature we examined.

To give some ideas, a sufficient condition for a part in music to be a pattern could be: if passage A = passage B, then A and B consist of occurrences of a musical pattern. Namely, only when two passages are an exact repetition of each other, they become a pattern. Another example might be that, if A = Transposition(B), A and B are two occurrences of a musical pattern. In other words, up to transposition, two passages are considered as occurrences of a pattern.

A very loose necessary condition could be: passage A and passage B have to have at least n interval(s) in common to be considered as belonging to a pattern. We may call this type of pattern intervallic patterns. A stricter necessary condition could be: passage A and passage B must at least have the same rhythm to be considered as occurrences of the same pattern. We may call this type of pattern metric patterns. Some datasets and algorithms follow the necessary condition that a passage has to be of length l to be considered a pattern. We believe these type of conditions can be useful in certain scenarios, but when used without attention to context and with unsuitable parameters, results can be limited in their fruitfulness.

Desiderata

By considering the existing definitions discussed above and our goals with pattern discovery algorithms, we can make a list of desired qualities of a pattern and a potential definition:

- Intuitive for musicians
- Computationally relevant
- Can be reused in multiple contexts (generalisability)

These two criteria may be difficult to meet. By having these two priorities, we highlight the scopes of pattern that we consider.

Patterns that fall in the scope of interest

As mentioned at the beginning of this chapter, we are interested in repetition and variations and their relations to patterns. They are also flexible, intuitive, and useful enough to ensure our desiderata. In later chapters, we will see that this leads us to the concept of transformation and use it to describe how one pattern occurrence connects to another.

We are also going to take excerpts of music as musical patterns from human annotations and algorithmic output. These excerpts can be viewed as a kind of exten-

sional definition for some humans and algorithms, and we would like to approximate them by using our intensional definitions.

These patterns are essentially what we find relevant to his dissertation, but they do not consist of an all-round definition. We cannot yet present our working definition of a pattern based solely on looking at other definitions thus far. We will examine a few additional aspects of pattern next, and hope to find a general enough pattern definition that would be as close as it can get to a sufficient and necessary condition at the end of this chapter.

Patterns that are not explicitly considered

There are reasons to give up certain types of pattern in our consideration and discussions from here on. Patterns from other domains, such as behaviour patterns and design patterns in *Object-oriented Programming (OOP)* will not be extensively discussed. There are similarities, however, such as the fact that "In object-oriented design, patterns are abstracted from the common structures that are found in software systems" (Zalta *et al.*, 2005). Further parallels could be drawn, perhaps in a separate line of work.

Closer to music, patterns of some specialised and artistic kinds, such as tension and resolution, are not explicitly considered. In addition, patterns specific to a composer or a genre are not explicitly studied. These two are listed because they play crucial roles in music analysis. The primary explanation for their exclusion is mainly based on a concern for generalisability. Another reason is to exclude factors such as biological factors, social background, and intentionality, which may come into play when we consider specialised patterns. For our final working definition, it is possible to include these patterns under a general scheme, but they will not be our primary focus.

The categorisation of patterns is not considered

One can categorise musical patterns according to different musical dimensions (Volk *et al.*, 2012). For example, one can differentiate between temporal patterns, pitch patterns, melodic patterns, and harmonic patterns. In this dissertation, we do not make hard distinctions between these categories. The reason for this is twofold. First, as our focus is on monophonic data, we do not have a strong tie with harmonic patterns, although we acknowledge that they are of extreme importance in many applications, as discussed in Section 1.5.1. Second, there is no guarantee that no

patterns lie in the overlap or the gaps between these categories. We do, however, concede that a taxonomy or ontology of musical patterns could give useful insight.

2.3 Musical patterns and related keywords

In previous sections, we have visited a variety of definitions of pattern. This section examines a range of terms associated with the concept of patterns, including segment, structure, motif, and so on. We will see their (dis)similarity to the word pattern, as well as create connections to some algorithms where these terms are used. The purpose is to explore and position ourselves and not to provide an overview of the algorithms we will examine in subsequent chapters.

2.3.1 In MIR and other algorithms

In this section, we introduce seven concept groups that are frequently encountered in MIR and other generic fields of related research. These are rough divisions, and in fact, it is not difficult to find analogous concepts in a variety of other areas.

Boundary

There are many symbols for marking boundaries in musical notation, such as the bar lines and repeat sign. There are also boundaries created by rests in the music, analogous to the punctuation in natural language. However, there also exist boundaries in music that are not written out; for example, within a bar of dense notes without rest, it is still possible for humans to demarcate one part from another, as we have shown in previous chapters, especially in Figure 1.1.

In relation to pattern and repetition, we largely concur with the statement that "patterns are defined primarily by virtue of repetition; cues such as surrounding silence, phrase boundaries, and measure lines are secondary factors that composers and performers can exploit to highlight or hide pattern occurrences." (Collins *et al.*, n.d.). Notated boundaries and boundaries that are not written out are indeed sometimes the boundaries of musical patterns. Conversely, given musical pattern instances, the beginnings and endings of these instances may serve as boundaries in the music.

In terms of applications, finding boundaries can be an important step in automating the editing, analysis, and creation of music, if not more. When one attempts to automate boundary detection in music, one could start by examining the boundaries directly or indirectly by examining musical patterns when they are available.

Segment and section

Both the terms "segment" and "section" in the music domain refer to a passage of music, with the conventions sometimes that sections are longer than segments. It is very common to spot segments and sections in music, just as in the visual domain: segments and sections of varying sizes can often be seen in virtually every composition of image and sound.

The segmentation task in [MIR](#) is the act of performing segmentation to break a whole into parts. As there has been no ongoing study on the "sectionalisation" task, we will focus on segmentation.

Performing segmentation in music is inextricably linked to the boundary detection and pattern discovery we discussed previously. While the boundaries of patterns may delimit their beginnings and endings, performing segmentation often does not distinguish between the beginnings and endings—the ending of a previous segment is the beginning of the next segment. Two more differences between pattern finding and segmentation are that:

- Unlike segmentation, the output of a pattern discovery algorithm or process does not always cover the entire piece of music, while segmentation normally does.
- Patterns can overlap with and nest within each other, while the segments do not usually overlap.

There are commonalities between the pattern discovery and the segmentation task in terms of the challenges faced by both. The boundaries of segments are very likely to correspond to the boundaries of patterns. Like the concept of pattern, the concept of segmentation also connects to music theoretic concepts such as figure, phrase, and section (Rodríguez López, 2016). As put in (Monelle, 2014), a pessimistic view about segmentation is that "Segmentation in music will always be ultimately based on intuition, because the relation of phonology and semantics, of expression and content, functions different in music." There have, however, been strides made on this front by the MIR community (Rodríguez López *et al.*, 2014; Wiering *et al.*, 2009). In addition, more discussion between the concept of pattern and segment can be found in previous work of musical pattern discovery algorithms such as (Lartillot, 2004; Velarde *et al.*, 2016).

Feature and viewpoint

A feature is a measurable property or characteristic of a phenomenon being observed (Bishop, 2006). This term is widely used in the context of machine learning and pattern recognition.

A related term in computational music analysis is the *viewpoint*, which is introduced by Darrel Conklin (Conklin & Anagnostopoulou, 2001) and can be viewed as a type of musical feature. Examples include interval and duration (Conklin & Anagnostopoulou, 2001). A viewpoint may be represented as a series of symbols. We tend to use the two terms interchangeably.

The important connection to patterns is that features may be used to determine patterns. It is often the case that patterns are viewed as patterns due to certain features they possess. For instance, Beethoven's 5th symphony is famous for its [unison, unison, unison, -3] intervallic pattern (Zbikowski, 2002).

A feature can be computationally learned from data or engineered based on domain knowledge. One of the benefits of leveraging deep learning models is that they circumvent the need to manually engineer features. In this dissertation, we will not use learned features, but rather predetermined ones. We will use feature analysis in subsequent chapters to demonstrate further connections between patterns and features.

Similarity, distance, and energy

In broad terms, (Hahn *et al.*, 2003) describes similarity as "a broad concept can be seen as an explanatory construct in the research area of memory retrieval, categorisation, problem solving, learning, linguistic knowledge, and processing, reasoning, as well as social judgement". Music similarity is a research topic on its own in MIR (Cambouropoulos, 2001; Park *et al.*, 2019; Volk *et al.*, 2012; Volk & Van Kranenburg, 2012). Various distance measures are employed together to investigate similarity (Janssen *et al.*, 2017), such as edit distance and earth mover's distance (Typke *et al.*, 2007). Taking inspiration from physics, one may think about the similarity and distance together with how much energy is needed to travel from one point to another. These concepts relate closely to musical patterns, as we will detail below.

(Reti, 1951) used four degrees of similarity to describe relations between musical patterns: imitation ("literal repetition of shapes, either directly or by inversion, reversion..."), variation ("changing of shapes in a slight, well traceable manner"), transformation ("creating essentially new shapes, though preserving the original substance"), and indirect affinity ("producing an affinity between independent

shapes through contributory features"). Other variations on these relations include relative repetition, ornamentation, substantive transformation (Serafine, 1988); and recurrence, development, response, contrast (LaRue, 1992). These degrees of similarity are, however, difficult to convert into a computational language.

Let us consider the connections with repetition and variation that might help with this situation. Similarity measures may determine whether there is a repetition to a certain extent. For example, having maximal similarity may be treated equivalently as having exact repetition, and having minimal similarity then corresponds to having heavily mutated variation. We will look at repetition and variation more closely in Section 2.3.2 in the context of music. More discussion about different aspects of similarity will come up in later chapters, too.

Through better understanding music similarity, we may peek into the wider importance of similarity in our understanding of other concepts in the world. Here, we move on to look at similarity in a broader sense, focusing on the divide between the intuitively understood notion of similarity by humans and the difficulty in converting it to computational systems.

Similarity is prevalent in our daily lives. Imagine playing the "spot the difference" game, in which the player is asked to find several places of differences between two largely similar images. Human players would normally not employ the strategy of scanning pixel by pixel, but rather by the object or the meaning of the image.

Similarity can help explain object recognition, category forming, conceptual knowledge structuring, behavioural prediction based on experience (Hebart *et al.*, 2020). (Goldstone & Son, 2012) listed four major classes of models that have been proposed for how humans assess similarities:

"In geometric models, entities are represented by their positions in a multidimensional space, and similarity is based on the proximity of entities in this space. In featural models, entities are described by their features, and the similarity of entities is an increasing function of their shared features and/or a decreasing function of their unique features. In alignment-based models, the similarity between two structured entities is calculated by placing the elements of their structures into correspondence. In transformational models, the similarity between two entities is conceptualized as the number of transformations required to transform one entity into the other. "

Indeed, the geometric models are linked to distances, allowing for the differences to be encoded in terms of distance. There are other measures such as the cardinality score, normalised matching score that may be used to calculate the distance. Energy

is important to consider, too: consider moving from one point to another—birds fly in straight lines, cars travel on roads, hikers might climb steep cliffs—energy is perhaps a better measure in these cases where the distance cannot capture what object is moving in what way. Moreover, energy is also a commonly used concept in machine learning models. In musical pattern discovery, energy, as a surrogate of distance, would translate to the consideration that we should be more aware of the different paths one musical pattern could take to be morphed into another one.

For the feature model, the features we discussed in the previous section may indeed be used to calculate similarity. For the alignment-based models, they have been computationally explored in MIR as well (Bountouridis, 2018). Transformation is something we are going to consider later in this dissertation (Chapter 5 and onwards).

Object, Gestalt, and grouping

In (Cook, 2000), the author looks at how Ligeti used a metaphor to explain his experience to "contain the sometimes impenetrable note-to-note patterning of the music within orderly bounds":

"One can imagine various objects in a state of total disarray in a drawer... The drawer too has a definite form. Inside it chaos reigns, but it is clearly defined itself."

Drawing a connection between finding musical patterns and physical objects is not difficult. The process of identifying a new pattern can be similar to the process of identifying a certain form of grouping and subsequently recognising it as an object. Later in this chapter, we will discuss the leitmotif, which is a musical pattern that may have a one-to-one correspondence with a certain object.

In daily life, finding an object and a certain grouping of several items is usually not difficult. There are, however, edge cases where objects and groupings can be alien, ambiguous, and fuzzy. Theorising these processes has seen many research efforts, including Gestalt psychology. There are six Gestalt principles (Wertheimer, 1938) of grouping: proximity, similarity, enclosure, common fate, continuation, closure, symmetry, past experience. There are also three key principles of reification, multistability, and invariance.

The principles have been applied in music research (Tenney & Polansky, 1980) as well as musical pattern discovery algorithms (Cambouropoulos, 2006; Lartillot, 2005; Velarde *et al.*, 2016). We will not dive into the details of each of these principles, but point out the fact that we specifically considered similarity, past experience (memory),

and multistability (ambiguity); we will consider symmetry and invariance together with transformation in the chapters after and including Chapter 5.

Structure, hierarchy, and heterarchy

Music is well-known for having rich hierarchical structures, ranging from global forms to local phrases, from harmonic progressions to melodic patterns. Repetition and variation have important roles to play: "musical structure often derives from repetition and is one of many crucial musical elements for defining and capturing structure" (Hunt, 2020). In (Ockelford, 2017; Wiggins, 1998), it has been noted that the meaning of music is in its structure, rather than being carried by its structure.

The dictionary definition of structure points us to other concepts such as arrangement, organisation, and mutual relation of the constituent parts. Once we have the grouping, objects, and patterns, another important ingredient is the way the structures are constructed on top of them. Additionally, structures may inform what kind of patterns exist inside, too.

In music, as well as several other cases, hierarchical structures are a common type of structure to see. Some examples include, in the visual domain, the pixel, edge, tex-ton, motif, part, and object hierarchy; in the language domain, the character, word, word group, clause, sentence, and story hierarchy. Figure 2.2 shows a simple example of how we could get to a deeper hierarchical structure with scales. Figure 2.3 shows a more complex example of a hierarchical analysis in Beethoven's composition.

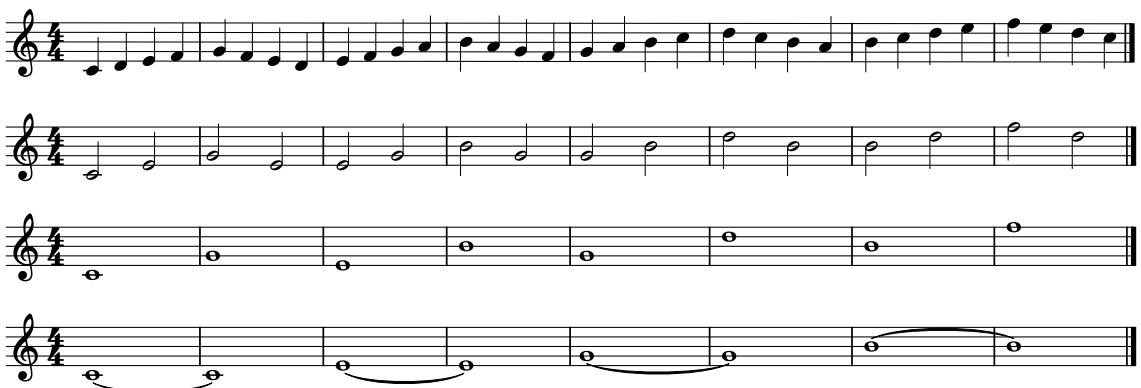


Figure 2.2: Scale interpreted with hierarchy. Four levels of hierarchy in an artificial example using scales: from the top to the bottom staff, we have different levels of details in different levels of hierarchy. The notes at different levels of hierarchy are specified based on the metrical positions of the notes.

Although the physical progression of time when one listening to music is linear, the important notes in music can form hierarchies subjectively. This happens to ex-

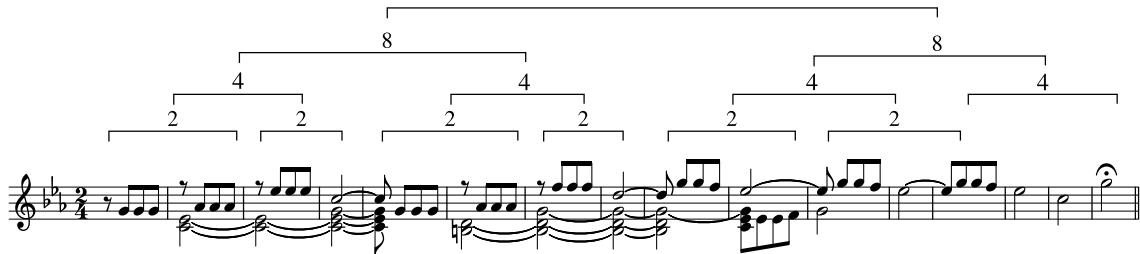


Figure 2.3: An example of hierarchical analysis. Lidov, 2005

perts as well as various kinds of listeners. The hierarchical structures allow us to examine music at different levels of detail and time scales. For example, in Figure 2.2, the patterns are shown on the top staff can be progressively and hierarchically reduced to the bottom staff.

There exist musical theories on this topic of the hierarchical structure of music, such as the *Generative Theory of Tonal Music (GTTM)* (Lerdahl & Jackendoff, 1985) and the Schenkerian theory of melodic reduction (Forte, 1959). The Schenkerian theory was described in an approachable way in (Cook, 2000), from which we can see the importance of reducing music complexity through hierarchy and its connection to musical patterns:

"Schenker did a reverse engineering job: he reduced it to a series of basic melodic and harmonic patterns, showing how these basic patterns were elaborated in the music Beethoven actually wrote ... it enabled you to understand the music in a way you otherwise couldn't. More specifically, it explained the moments of apparent incoherence as purely superficial phenomena resulting from the elaboration of the underlying structure; it allowed you to hear 'through' the surface to what lay underneath."

Structures do not have to be hierarchical. Heterarchy, first coined by McCulloch (1945), has been used to describe cognitive structures and to understand social organisation. It expands on hierarchy and emphasises the relation of elements. In a heterarchy, the relation between elements possesses the potential for being ranked in a number of different ways or being completely unranked (Crumley, 1995). In other words, instead of tree-like relations in the case of a hierarchy, we may also have a model from graph theory or complex network theory, where there are simply nodes and links with no hierarchy.

In MIR, structural analyses (Allegraud *et al.*, 2019; Nieto, 2015) have been made and tools have been developed for visualising structures (Müller & Jiang, 2012). Some structures are also known as musical forms, which have been subjected to algorithmic analyses (Giraud *et al.*, 2016).

In connectionists' algorithms, learning structures and meaning has been a struggle. For example, an image containing a shirt with a picture of a pretzel on it would be recognised as an actual pretzel, and musical structures created by generative models also tend to be local (not more than a few bars long) (Kortylewski *et al.*, 2021; Wu & Yang, 2020).

Structure is also an important aspect for implementing algorithms. Program structure and algorithm architecture have their own research areas. We consider these aspects more in Chapter 5, and we do not discuss them further here as there is not always a parallel between these structures and the structures in music.

Grammar and rule

Grammar is very closely related to structure, especially hierarchical structures, because grammar may be used to create and summarise such structures. Grammar can be thought of as a system of rules and principles. It is known that various aspects of cognition are shaped and enriched by abstract rules, which help to describe, link, and classify events into meaningful concepts (Mansouri *et al.*, 2020). In language, (Blacking, 1984) argues that "Grammars are attempts to codify the regularities of structure that communities generate in order to give coherence to their communication and to enable individuals to share meanings." In music, one seminal work about grammar is [GTTM](#).

According to [GTTM](#), a listener unconsciously infers four types of hierarchical structure in a musical surface (as summarised in (Pearce, 2005)):

"

- first, grouping structure which corresponds to the segmentation of the musical surface into units (e.g., motifs, phrases, periods, and sections);
- second, metrical structure which corresponds to the pattern of periodically recurring strong and weak beats;
- third, time-span reduction which represents the relative structural importance of pitch events within contextually established rhythmic units;
- fourth and finally, prolongational reduction reflecting patterns of tension and relaxation amongst pitch events at various levels of structure."

The patterns we consider go up to at least the third level, as we previously stated that we do not consider the tension and relaxation pattern per se. We will discuss motifs

and phrases in more detail later in this chapter. Metrical aspects are considered with feature analysis in the chapters to come. However, we do not consider many other aspects of hierarchy and grammar in this dissertation. We treat musical patterns the same way regardless of where they lie in different levels of hierarchy, because there is little research on devising and evaluating algorithms that are capable of extracting a hierarchy or grammar and thereafter using them to produce musical patterns.

The importance of grammar and [GTTM](#) is evident in a variety of areas of research, including MIR. A series of computational methods have been developed for creating and modifying musical pieces according to GTTM (Hamanaka *et al.*, 2014, 2015). Additional forms of grammar have also been seen in the research scene at the intersection of music and computation (Melkonian, 2019; Young, 2017). In (Wiggins, 1998), another type of grammar was applied in the context of music: "pattern grammars are a specialised grammatical formalism which extend standard Chomskian grammars". A pattern in the pattern grammar is a non-empty finite string of symbols. (Angluin, 1988) demonstrates how it functions as follows:

"The language of a pattern p , denoted $L(p)$, is the set of all strings over the alphabet A obtained by substituting non-empty strings of constant symbols for the variable symbols of p . If $p = 122x5yyx3$, then the language of p includes the strings 12205111103 and 122001512120013, but not the strings 12253 or 1221560601113."

This may remind us of regular expressions in a computational context. We will focus on a related topic—querying patterns—in later chapters.

2.3.2 More music-specific

In the previous section, we have seen connections between pattern and other concepts in MIR as well as adjacent fields. This section covers twelve more concepts that are more music-specific, which implies that they are primarily from the music analysis tradition, and we will see more literature from this area.

Theme, phrase, repetition, and variation are common terms not only seen in music but also in other domains. In this section, we consider them largely in the context of music. Where necessary, we also consider their connections to the algorithmic aspects.

Several of these concepts in this section are often hard to generalise. They are often associated with a certain composer, style, or genre. The term pattern has been used as an umbrella term for many of them. For example, in the original [MIREX](#) pattern discovery task, the full name of the task expands the word pattern into three other

terms: motif, theme, and section. The definitions of these three terms are given below by the MIREX task:

According to Drabkin (2001a), a "motif may be of any size, and is most commonly regarded as the shortest subdivision of a theme or phrase that still maintains its identity as an idea."

A theme is the "musical material on which part or all of a work is based, usually having a recognizable melody and sometimes perceivable as a complete musical expression in itself" Drabkin (2001b).

A repeated section is the "restatement of a portion of a musical composition of any length from a single bar to a whole section, or occasionally the whole piece. Since the Classical period, repeated passages have not usually been written out; instead they are enclosed within the signs ||: and :||" (Tilmouth, 2001).

There are more works from others who used different words for analysing music. For example, Schoenberg wrote in (Schoenberg, 2006) that components of a piece include "statement, phrase, gestalt, motive, feature, grundgestalten, and figure". However, it is unclear as to how these divisions connect with one another (seven concepts generate $7 \times 6 = 42$ relations to consider, if not considering one-to-many and many-to-one relations), and how they may aid in understanding music and devising computational methods.

We do not claim that specific patterns and terminologies do not matter and cannot provide helpful insights. We do believe that a generalised concept can help with unifying the separate concepts. For example, in (Sears & Widmer, 2020), the authors try to discover voice leading patterns and shed new light on specific polyphonic patterns in music. In the same work, it has been noted that "the computational music analysis community could not only improve upon the current pattern discovery pipeline, but perhaps more importantly, develop a more sophisticated theory about the organizational principles that characterize recurrent patterns in music, cadential or otherwise." We concur with this viewpoint.

Although we do not develop this theory of organising principles for musical concepts, we believe that many of these specific concepts can be unified into the topic of pattern discovery. In fact, the concept of pattern has already been used to subsume other concepts in music, which is the case with the MIREX task. In addition, according to (Simon & Sumner, 1993), "one of the purposes of analyzing musical structure and form is to discover the patterns that are explicit or implicit in musical works", demonstrating the unification capacity of patterns once more. In the rest of this sec-

tion, we start with the broad concepts of repetition and variation and progress into more specific concepts.

Repetition and Variation

Musical patterns are closely related to repetition and variation. It is often the case that repetitions anchor patterns in musical compositions, followed by the development of patterns under variations, with the aim of creating appealing musical structures. In fact, we have mentioned repetition and variation a few times so far, especially in the overview of this chapter. Let us dive into this topic again and see how others have written about them and consider their connections with musical patterns. We will examine ideas from a variety of scholars on this subject to reinforce these connections and expand our perspective.

The importance of repetition and variation is widely agreed upon. (Lerdahl & Jackendoff, 1983) observed that "the importance of parallelism (repetition) in musical structure cannot be overestimated". (Rahn, 1993) even goes as far as to say that "all musical structure derives from repetition". For Schenker (Schenker, 1980), "only by repetition can a series of tones be characterized as something definite. Only repetition can demarcate a series of tones and its purpose". In her seminal book (Margulis, 2014), a thorough examination of the function, importance, perception, and many other aspects of repetition are presented. The book provides analysis of and insight into repetition in a variety of contexts, including evolution, language, music, and dance. In other contexts, (Fitch, 2006; Margulis, 2014) both suggested that the prediction of pattern repetition distinguishes music from language more than any other feature. More on the point of variation, in (Zbikowski, 2002), it was noted that "the motive undergoes continual change, as the different versions". (Schoenberg, 1967) commented on how repetition is not enough with variations: "Repetition alone often gives rise to monotony. Monotony can only be overcome by variation".

Repetition and variation as concepts are, however, not without their own complications of vagueness, as once put by Meyer: "I fully agree with Tovey that 'Nothing is easier than to derive any musical idea whatever from any other musical idea'" (Meyer, 1973). Another point in (Mazzola, 2018) concerns the relation between repetition and variation: "if we include variation as a kind of repetition, we must acknowledge immediately that variation clouds this distinction between abstract and concrete domains of form". Variation always include an element of repetition, but if the variation is sophisticated, its repeated aspects may be perceptible only to the listener who already knows its style.

We have also touched on these topics in previous sections. Summarising them all, there seems to be a need to concretise various forms of repetition and variation. In this dissertation, we acknowledge the validity of the approach of treating inexact and exact repetition as different types of repetition; however, we treat inexact repetition as variation, not as repetition, as we mentioned in Figure 2.1, at the beginning of this chapter. We also separate out the issue between endogenous (concerning data) and exogenous (concerning perception) variations; if two bars of music appear subtly different on sheet music but sound the same to some people, we regard these two bars of music as a variation rather than repetition.

Idea

The concept of musical idea has been discussed extensively in books and articles (Ferguson, 1941; Hanninen, 2003; Schoenberg, 2006). (Ferguson, 1941) explores what a musical idea is starting from the definition that "An idea is that once an image and a valuation of experience". (Brand, Hailey, *et al.*, 1997) reiterate Schoenberg's view on musical idea and composition "Composing is: thinking in tones and rhythms. Every piece of music is the presentation of a musical idea (Idee, Gedanke, Einfall)". For Schoenberg, "A musical idea is sheerly musical. It is a relation between tones." (Schoenberg, 2006). We can immediately observe some misalignment in the use of the word "idea".

(Hanninen, 2003) introduces the concept of recontextualisation using musical ideas and their instances. An idea was defined as "a set of one or more contextual (not sonic or structural) criteria. Ideas have (and manifest in) instances." Contextual criteria here means a set of criterion that

"identifies a characteristic of a grouping with a propensity for association among groupings within a musical context under consideration. Contextual criteria are activated by repetition; they indicate equivalence or similarity in non-linear musical spaces including pitch contour, duration series,..."

This definition is more in line with what we have seen so far about repetition and musical patterns. Schoenberg has also commented on the important role played by repetition for musical ideas: "Repetition is one of the means (in presenting an idea) to promote the comprehensibility of the idea presented." (Schoenberg, 2006).

We can see from this body of literature that a musical idea is analogous to a musical pattern: both are derived from music and our subjective experiences, are closely related to repetition, consist of tones and rhythms, and are associated with a set of

criteria and occurrences. Next, we will examine the musical unit, figure, and other concepts that build on top of them.

Unit, figure, phrase, and more

In (Goetschius, 1904), a range of concepts have been discussed, including unit, idea, figure, motif, phrase, sentence, and more:

"The smallest unit in musical composition is the single tone. The smallest cluster of successive tones (from two to four or five in number) that will convey a definite musical impression, as miniature musical idea, is called a Figure. Assuming the single tone to represent the same unit of expression as a letter of the alphabet, the melodic figure would be defined as the equivalent of a complete (small) word, pursuing the comparison further, a series of figures constitutes the melodic motive, equivalent to the smallest group of words (a subject with its article and adjective, for example); and two or three motives make a Phrase, equivalent to the complete, though comparatively brief, sentence (subject, predicate, and object)."

We agreed that the smallest unit is a single musical event. We would argue that other concepts themselves are also different types of units, and we have a range of concepts that are made of the composite of different units. This succession of concepts seems to get gradually longer, but without predefined numerical intervals of length. The source of inspiration seems to be natural languages, which adheres to conventions such as punctuation and capitalisation. These conventions are, however, not standardised in music. A long sequence in a short musical composition might be a short sequence in a long musical composition. One can imagine the difficulties in arguing that a certain series of notes does not constitute a figure but a motif.

Moreover, some of those definitions can be in conflict with each other from different literature. For example, a phrase is defined in ("Phrase", 2001) as

"A term adopted from linguistic syntax and used for short musical units of various lengths. A phrase is generally regarded as longer than a Motif but shorter than a Period. As a formal unit, however, it must be considered in its polyphonic entirety, like 'period', 'sentence' and even 'theme'."

Period was not mentioned in the previous quote at all when we gave the first definition from a different source in this section. Additionally, "short musical units of various lengths" ("Phrase", 2001) can hardly be equated with "two or three motives" (Goetschius, 1904). The intrinsic difficulty is that one musical unit, figure, or phrase can cease to be a musical unit, figure, or phrase in a different musical context. We

will, therefore, not further distinguish between the terms in this series of concepts for this dissertation.

The same issues of context and length exist with musical patterns. The context and the relative length of the pattern is difficult to describe formally and comprehensively. We will circumvent these issues by using transformation in later chapters, but they remain difficult issues in and of themselves. Next, we will discuss themes and different types of motif on their own.

Theme

We have already seen the keyword "theme" appearing in previous sections. Here, we give a few more viewpoints and relevant research on musical themes.

For Schoenberg (Schoenberg, 1967),

"The formulation of a theme assumes that there will follow 'adventures,' 'predicaments,' which ask for solution, for elaboration, for development, for contrast"

In jazz music, a theme is

"The musical material on which part or all of a work is based, usually having a recognizable melody and sometimes perceivable as a complete musical expression in itself, independent of the work to which it belongs."

(Drabkin, 2001b; "Theme (jazz)", 2003)

These descriptions are very difficult to serve as specifications for computational methods. There are, however, efforts to gather and digitise data of musical themes (Zalkow *et al.*, 2020) so that perhaps a machine learning method could help with reifying what recognisability means in music.

Not unexpectedly, musical themes also have a close relation with repetition and variation: "On the one hand, the musical theme asserts itself, but on the other, the theme extends outside itself." (Kaduri, 2006). We can also take the words such as "development" and "contrast" in Schoenberg's quote and treat them as alluding to variations. From this connection with repetition and variation, in combination with how themes were included in the [MIREX](#) task, we treat themes as a type of pattern in this dissertation.

Themes are sometimes considered longer than patterns, but a hard cutoff point is difficult to establish, as we discussed in previous sections. We do acknowledge that a pattern that constitutes a theme differs from a pattern that merely provides texture (Utgoff, 2006). This finer categorisation of patterns is not included in the scope of this dissertation.

Motif

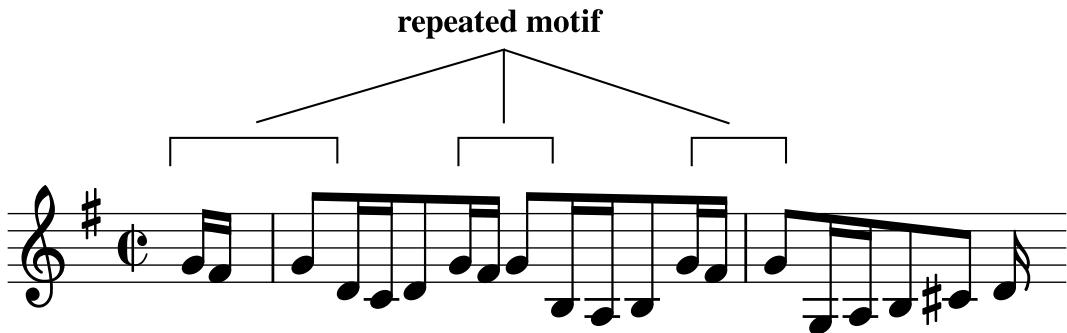


Figure 2.4: Example of a repeated motif in Bach's Brandenburg. Reproduced from (Ockelford, 2018).

In Figure 2.4 (Ockelford, 2018), we show an example of motifs. In (Drabkin, 2001a), a motif is defined to be

"A short musical idea, melodic, harmonic, rhythmic, or any combination of these three. A motif may be of any size, and is most commonly regarded as the shortest subdivision of a theme or phrase that still maintains its identity as an idea."

Schoenberg's (Schoenberg, 1967) definition seems to agree with this and adds on to the definition by emphasising the time offset at which a motif appears:

"The motive generally appears in a characteristic and impressive manner at the beginning of a piece. The features of a motive are intervals and rhythms, combined to produce a memorable shape or contour which usually implies an inherent harmony. A motive appears constantly throughout a piece: it is repeated."

For Réti (Buteau & Mazzola, 2008), a general motif is:

"... any musical element, be it a melodic phrase or fragment or even only a rhythmical or dynamical feature which, by being constantly repeated and varied throughout a work or a section, ..."

In addition, Réti is unconcerned with accurate nomenclature because he writes subsequently that "In general, the author does not believe in the possibility or even desirability of enforcing strict musical definitions.".

There have been concerns about these definitions and views. For example, in (Mazzola, 2018), it has been pointed out that

"Réti asserts that he handles the words 'motif' and 'theme' without categorical distinction, but a difference between the two concepts is that a theme as a fuller group or 'period' which acquires a 'motivic' function in a composition's course."

Concerning transitivity, (Buteau & Mazzola, 2000) commented that "Reti's concept of identity of motif shapes includes relations—such as variation, which is a kind of similarity—which are not necessarily transitive."

Let us now take a look at these definitions and concerns as a whole. Through the appearance of repetition and variation in the definitions, we see a connection with the concept of pattern. Like patterns, we also observe some disagreements and concerns about how motifs are defined. Some of the arguments are closely related to what we have discussed previously in terms of context and lengths. For transitivity, we do bear in mind that if one passage is varied multiple times, it could become hard to relate the last occurrence back to the initial occurrence. For a definition, we do not see a risk in treating motifs as another type of pattern. In fact, with Réti's definition, we can use pattern and motif interchangeably.

Algorithms have been devised (Ganguli *et al.*, 2017; Jiménez *et al.*, 2011) to discover motifs automatically. However, they have not been submitted to the [MIREX](#) pattern discovery task; hence they are not our focus in this dissertation.

Leitmotif

As a special type of motif, leitmotifs can be seen in the compositions by Wagner, in films, and in other background music. In (Whittall, 2001), it is defined as

"In its primary sense, a theme, or other coherent musical idea, clearly defined so as to retain its identity if modified on subsequent appearances, whose purpose is to represent or symbolize a person, object, place, idea, state of mind, supernatural force or any other ingredient in a dramatic work."

It seems this definition largely agree with the definition in (Ockelford, 2018), which defines a leitmotif to be "A characteristic melodic, harmonic, or rhythmic idea that is associated with a particular person, place, emotion or idea." Leitmotif is a special kind of pattern since it has a semantic meaning, which most patterns do not possess.

In a computational context, datasets on leitmotifs exist, but are not yet well-researched computationally. We do not focus on leitmotif in the dissertation, though it is a potential application direction of our research.

Reminiscence and head motif

To further enrich the "motif" family, we see two more types of motif in this section. In ("Reminiscence motif", 2002), a reminiscence motif was also defined using theme, musical idea, and has a historical connection with leitmotif:

"A theme, or other coherent musical idea, which returns more or less unaltered, as identification for the audience or to signify recollection of the past by a dramatic character. It is an important ancestor of the Leitmotif."

In (Fallows, 2001), a head motif was defined with musical idea once again. It is a more constrained version of the motif:

"A musical idea which by virtue of appearing at the beginning of each of a series of pieces or movements establishes a relationship between them."

The definitions of these particular types of motifs demonstrate once again how definitions interact and rely on one another, posing difficulties for computational methods. Admittedly, when the historical provenance and more specific descriptions are given, such as those definitions above, it is arguably more straightforward to devise an algorithm to automatically find them. For example, finding a head motif should be easier than finding motifs because the search space will be significantly reduced. We do not yet know of any algorithms targeting these types of motif.

Ostinato

(Schnapper, 2001) defines ostinato as below:

"A term used to refer to the repetition of a musical pattern many times in succession while other musical elements are generally changing."

In the same article, the importance of rhythm is also stressed: "The regular repetition of a pattern requires, as a minimum, the existence of a rhythmic structure, to which other elements may be added." (Schnapper, 2001). It is also mentioned that ostinatos can be found in oral traditions, jazz, Baroque, minimalist, and popular music, whereas a decline occurred in classical and romantic eras. Additionally, the ostinato also plays an important role in musical structure and expression.

In connection to the concept of pattern, we see that musical pattern is used to define this term and that repetition and variation also play a significant role. Ostinato and pattern share many similarities, and we deem that it is synonymous with pattern. We cannot find any algorithms targeting the discovery of ostinatos in music.

Lick

In (Witmer, 2001), a lick is defined as

"A term used in jazz, blues and pop music to describe a short recognizable melodic motif, formula or phrase. Improvising jazz and blues musicians have at their disposal a repertory of licks, some of their own invention by which they can be identified, some borrowed from other players, and a solo may be little more than the stringing together of a number of such fragments."

This term seems to be the jazz equivalent of a motif or a pattern. We can expect, however, many specificities of patterns in jazz music may come from this different improvisational context. There have been research concerning jazz-specific patterns and algorithms (Klaus Frieler & Dixon, 2018; Quick & Thomas, 2019). This dissertation does not have an emphasis on jazz music, but we do use a jazz dataset in a later chapter.

Schemata

In a separate vein of Gallant music, we see some seminal work on schemata (Gjerdingen, 2007, 2014), which are described to be "stock musical phrases" (Gjerdingen, 2007). Figure 2.5 depicts an abstract form of a schema—Romanesca, while Figure 2.6 shows an example of this schema in actual music. Figure 2.7 shows a complete analysis of schemata in Haydn's variations.

Although schemata are primarily concerned with polyphonic music, we provide several examples here for three reasons. First, in later chapters, we will use music from this era and will not discuss the schemata theory there. We do think that schemata are a type of musical pattern, but we do not discuss them further in this dissertation due to their polyphonic emphasis.

Second, these examples demonstrate the extra complexity of polyphonic musical patterns, which necessitates the assistance of computational methods. In fact, we will use some non-schema based polyphonic musical pattern discovery algorithms in later chapters. The schemata may serve as a useful reference for these algorithms. Additionally, there are efforts towards computationally modelling the schemata theory, such as in (Finkensiep *et al.*, 2020) and (Katsiavalos *et al.*, 2019).

Third, in Figure 2.8, we see a formulation of seeing music as a string of concatenating and overlapping beads made of schemata. The music can branch off (M, N, O, P in the figure) or choose from different possibilities (A, B, C). This is how we see music can be structured with patterns as well.

2 Pattern and Pattern Discovery

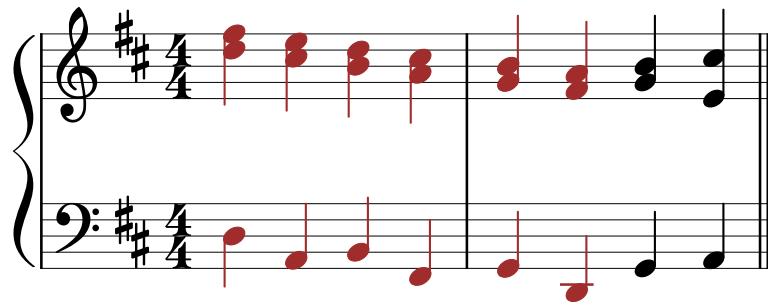


Figure 2.5: A schemata: Romanesca. Reproduced from Gjerdingen, 2007.

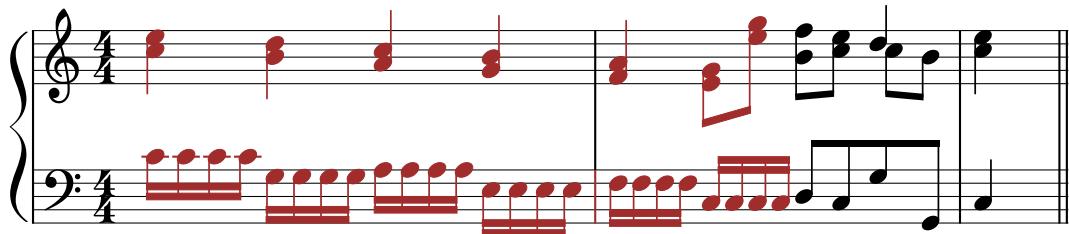


Figure 2.6: Romanesca in music with rhythmic elaboration. Reproduced from Gjerdingen, 2007.

134 MUSIC IN THE GALANT STYLE Chapter 10 A THEME AND VARIATIONS BY HAYDN 135

Var. I

Var. II

Figure 2.7: Analysis on Haydn's Variations based on schemata theory, showing the complexity and possibility for automation. Taken from Gjerdingen, 2007.

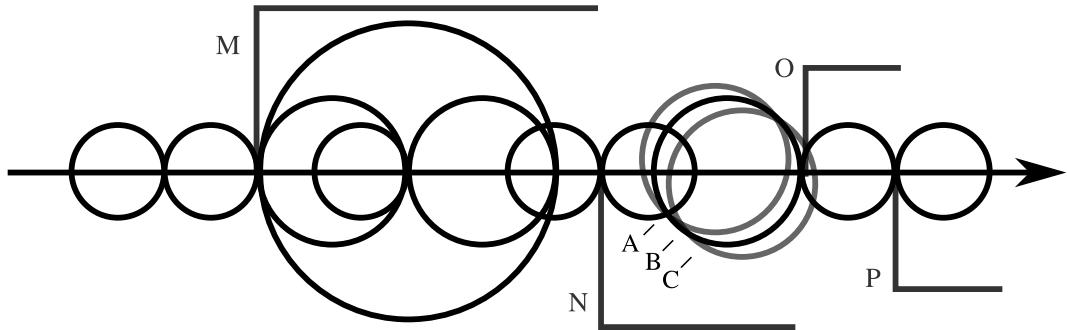


Figure 2.8: Composition of schemata. Reproduced from Gjerdingen, 2007. Music can be seen as a series of "beads". They are sometimes branching and overlapping possibilities.

Gregorian Neumes

The last type of music we are going to examine is Gregorian chant. The notation unit for Gregorian chants goes beyond a single note.

For example, the scandicus flexus is a neume that consists of four notes, going up and then dropping down, as in F, G, A, F (Kephart, 2017). Climacus is another neume that consists of three or more downward-moving notes.

Although we are not listing all neumes here, we can already see that these two can be referred to as musical patterns. Neumes are actually called neume patterns occasionally. Once translated into the modern staff notation, to discover the patterns would be a reverse-engineering problem, that is, converting the notes back to the neumes. Computational pattern discovery methods have been used to generate melodies for lost chants (Conklin & Maessen, 2019).

2.4 Pattern discovery algorithms

Leaving behind various concepts from previous sections, we focus on introducing pattern discovery algorithms in this section. Algorithms have been devised to discover patterns using mathematical principles and *heuristics* from the music domain. They take musical pieces as input, and output a set of patterns occurring within the piece.

In this section, we provide more information for *SotA* algorithms. In particular, we introduce the musical pattern discovery algorithms that we use for our experiments later in Chapter 4. We also introduce several algorithms designed for pattern discovery in other specific domains as well as some generic pattern discovery algorithms that function across domains.

2.4.1 Musical pattern discovery algorithms

Based on the availability and compatibility of the algorithms, we examine and introduce 11 musical pattern discovery algorithms, including two families of algorithms that include 7 more algorithms in total. Out of these algorithms, a few have been submitted to the MIREX task. Each algorithm will be accompanied by a short description.

The readers do not have to remember all details of the algorithms to interpret the results we deliver in this and subsequent chapters. Our focus is to look at the algorithmically discovered patterns themselves more than the mechanism of algorithms, which is not uncommon (the philosophy behind MIREX is similar) and has the potential to be more generalisable. More specially, we treat these algorithms as a black box for two reasons. First, there are more and more black-box algorithms developed from machine learning, especially deep learning. Second, even with the white- and grey-box algorithms, even when there are mathematical models behind these algorithms, the actual implemented operations on data may surpass our mental ability to track the application of them in real data, which also generates the need to treat the algorithms as modular, information-hiding black boxes.

Wavelet-based (Velarde *et al.*, 2016)

Wavelets and related applications have a strong theoretical foundation in mathematics and physics. By convolving wavelets with symbolic melodies, we obtain correlations between the unknown symbolic melodies and the known patterns encoded in the wavelets. The resulting correlations are used for melodic segmentation and concatenation data, which is then compared, clustered, and ranked. These ranked clusters then become the melodic patterns retrieved from the piece. The two algorithms, [VM1](#) and [VM2](#), have two Matlab implementations using several different parameters. From the original paper, we know that according to Friedman's test, VM1 and VM2 show no significant difference in the results of the three-layer F1 score, a metric in MIREX (will be discussed in Section 4.2.1); and for discovering exact occurrences, VM1 outperforms VM2.

Compositional Hierarchical Model (Pesek *et al.*, 2017)

The [Symbolic Compositional Hierarchical Model \(SYMCHM\)](#) model was inspired by object categorisation in computer vision. As an alternative to deep learning models, the model imposes a hierarchical structure in different compositional layers and tries to provide a transparent deep architecture to discover repetitions in music. The

method is based on the assumption that repetitive patterns can be characterised by the number of occurrences of their sub-patterns. As a consequence, the model learns pitch patterns from symbolic data in an unsupervised manner.

According to the original paper, the process of inference in SymCHM is designed to be either exact or approximate. To approximate, two processes, hallucination and inhibition, are involved in finding the parts that have changed, inserted, or deleted.

Geometric Algorithm Family

The name geometric comes from the fact that the patterns in music are found by identifying their shapes in sheet music. This family of geometrically inspired pattern discovery algorithms that can identify note groups with similar shapes and therefore extract these shapes as musical patterns. In the design of these algorithms, factors contributing to the perceived importance of a pattern were discussed. The factors identified include the compactness (the notes in the pattern are contiguous) and the amount of compression that can be achieved, which has a link to repetition and variations.

For the algorithms in this geometric family, an important concept is that of [Maximal Translatable Pattern \(MTP\)](#),

$$MTP(v, D) = \{p | p + v \in D \wedge p \in D\}$$

where p denote pattern, D denotes a dataset, and v a vector. MTP for a vector v contains all points mapped by v onto other points in the dataset

Another important and related concept is that of [Translational Equivalence Class \(TEC\)](#),

$$TEC(P, D) = \{Q | Q = P + v \wedge Q \in D\}$$

where P and Q denote patterns, the rest similar to those in [MTP](#). TEC is the set of all translationally invariant occurrences of a pattern.

This family of algorithms is designed with polyphonic pieces in mind. Six algorithms, [SIA, Structure Induction Algorithm for TECs](#) [Meredith et al., 2001 \(SIATEC\)](#), [Data COMpression using SIATEC \(COSIATEC\)](#), [SIA with a sliding window of size r \(SIAR\)](#), [SIATECCompress-Precision \(SIACP\)](#), [SIATECCompress-Recall \(SIACR\)](#), [SIATECCompress-F1 \(SIACF1\)](#), [SIA for r superdiagonals and Compactness Trawler-Categorisation and Fingerprinting \(SIACFP\)](#), and Forth, all belong to this family of geometric musical pattern discovery algorithms. A tool written in Java is available online, supporting this array of point-set compression

2 Pattern and Pattern Discovery

algorithms. We provide a rough overview of this family of algorithms below; please see (Meredith, 2015) for a comprehensive review.

SIA and SIATEC (Meredith *et al.*, 2002)

The SIA algorithm calculates all vectors between all notes. Along with using a couple of sorting procedures, the algorithm discovers all MTP. As a result, SIA admits unlimited gaps in its pattern.

Proposed in the same paper as in the SIA algorithm, SIATEC computes the TEC based on SIA. The algorithm computes all the occurrences of each of the maximally repeated patterns discovered by SIA.

SIA only finds one occurrence of each MTP and one vector by which that MTP is translatable within the dataset. SIATEC, on the other hand, finds all the non-empty MTPs and then all the patterns within the dataset that are translationally equivalent to each MTP.

COSIATEC and SIATECCOMPRESS (Meredith, 2013)

Both COSIATEC and SIATECCOMPRESS iteratively apply the SIATEC algorithm to compress a piece into the corresponding union of TECs of MTPs. The difference is that COSIATEC finds the best TEC and then removes its cover set from the input, while SIATECCOMPRESS extracts a list of MTP TECs and then selects a subset of the best TECs to cover the input. The selection criteria are based on compression ratio and compactness measures.

The SIATECCOMPRESS algorithm comes with three variants with different parameters. These different parameters were obtained by optimising for precision, recall, and F_1 score.

Forth (Forth, 2012)

Based on the geometric method SIATEC, the Forth algorithm incorporates more considerations based on music theory and cognition. It employs heuristic measures in conjunction with compression ratio and compactness value thresholds. To list the differences between the three algorithms mentioned above:

- COSIATEC finds patterns that are exclusive (no overlapping patterns) and exhaustive (every note belongs to a pattern).
- SIATECCOMPRESS and Forth's algorithm generate patterns that can intersect, which makes them faster but less effective at compression than COSIATEC.

SIAR (Collins, 2011)

Based on the [SIA](#) algorithm, [SIAR](#) limits the creation of vectors from all possible note combinations by constraining them to a maximum of $R \in \mathbb{N}$ successive notes. In this way, we might think of R as the "memory size" parameter specifying the number of notes to consider in [SIA](#).

SIARCT-CFP (Collins *et al.*, 2013)

This algorithm combines [SIA](#) with a *compactness trawler*, fingerprinting, and categorisation steps. A compactness trawler removes all patterns that do not satisfy a parameterised compactness ratio $c \in \mathbb{R}$ —the ratio between the number of notes in a pattern and the width of the bounding box of the pattern. The fingerprinting step computes the rhythmic ratios in order to allow rhythmic variations. The categorisation step ranks the discovered patterns in order of importance. [SIACFP](#) algorithm has an implementation in Matlab.

Pattern Clustering (de Reuse and Fujinaga, 2019)

Leaving the geometric algorithms behind, a recent pattern discovery algorithm uses a clustering approach. This algorithm maps passages of music onto a low-dimensional subspace using an embedding learned from human annotations of repeated patterns. The neural network used for learning the embedding is a fully connected, feedforward network with two hidden layers, each containing 100 nodes. Techniques such as dropout and batch normalisation were used, with an output layer size 5.

MGDP (Conklin, 2010)

The [Maximally General Distinctive Pattern \(MGDP\)](#) algorithm computes the maximally general distinctive patterns in a corpus. The algorithm is a sequential pattern mining method and works by constructing a suffix tree, which is then pruned based on the patterns' occurrences.

Closed pattern mining

[Closed patterns](#) are the longest patterns that are repeated the same number of times. Finding closed patterns is a standard technique in data mining. A closed pattern mining algorithm can be used to find patterns with pitch intervals and rhythmic ratio and have been used in (Lartillot, 2005; Ren, 2016).

PatMinr (Lartillot, 2014)

The **Pattern Miner (PatMinr)** algorithm considers both the concept of closed pattern and cyclic pattern. It uses an incremental one-pass approach to identify pattern occurrences, considering various music attributes such as chromatic, diatonic pitch, metrical position, and articulation. A pattern prefix tree is employed in the implementation. The algorithm is available in Matlab as part of the **MIR** toolbox (Lartillot, 2011). This algorithm has been submitted to the MIREX task with two different sets of parameters.

FlExPat (Rolland, 1999)

This algorithm uses Levenshtein edit distance to compare two passages, construct a similarity graph, and then identify them as instances of the same pattern. It considers local and global music aspects such as absolute pitch, backward interval, crescendo, contour, time signature, tempo, and so on. The algorithm is built into the Imrology software system (Rolland, 1998).

MotivesExtractor (Nieto, 2015)

MotivesExtractor employs perceptual grouping principles. It computes a distance matrix between pairs of potential motives and then uses a clustering algorithm to identify the instances of the same pattern. The implementation of this algorithm is in Python.

In more detail, the algorithm obtains a harmonic representation of the audio or symbolic input and extracts patterns based on a produced **self-similarity matrix**. (Nieto & Farhood, 2012b) Each number in the matrix denotes a similarity value of a pair of elements in the sequence, with the diagonal being identical. With some post-processing steps, pairs of high similarity are extracted and are considered to be the occurrences of the same pattern. Based on Gestalt principles, the algorithm also filters out the patterns that contain relatively long notes or rests, or relatively large intervals.

PAT (Cambouropoulos, 2006)

PAT algorithm employs Crochemore's set partitioning method to recursively split the melody into sets of repeating tokens. As a result, **maximal patterns** are found and filtered according to pattern lengths. The algorithm was used for segmentation based on maximal repeated patterns.

Suffix arrays ((Knopke & Jürgensen, 2009))

In (Knopke & Jürgensen, 2009), suffix arrays were used to represent phrases of Palestrina masses. To extract patterns, the algorithm first uses rests as an indicator to find phrases, and inserts each phrase four times (normal order, inversion, retrograde, and retrograde inversion) into the new representation and matches the repetition.

2.4.2 Generic pattern discovery

Turning to generic pattern discovery algorithms, the recent advances in multi-purpose pattern discovery algorithms have been shown to be useful in many use cases, including natural language processing, signal processing, computer vision, and so forth (Bishop, 2006). One underlying reason is that many of the algorithms are based on information-theoretic and statistical principles, which gives rise to the prospect of high generalisability.

Although it could be a success to use one of these general-purpose algorithms directly, musical pattern discovery faces many unique challenges, as described previously. The context-free copying of a methodology runs the danger of resulting in a naive, wholesale import. From a more algorithmic perspective, for example, problems such as pitch-spelling, key-finding, local tonality, temporal context switching are all highly domain specific problems we face when trying to discover patterns from music. In fact, several algorithms we introduced in the last section have generic components and adaptations to the music domain. In this section, we briefly examine a broader range of possibilities from generic pattern discovery methods and discuss their suitability for discovering patterns in music.

Data mining algorithms

Data mining algorithms intend to uncover inherent regularities in datasets (Han *et al.*, 2011). There are five major components (Saxena & Rajpoot, 2009) in this area of research: "association rules, classification or clustering, characterization or comparison, sequential pattern analysis, and trend analysis". From this area of research, we have seen definitions such as **maximal pattern** and **closed pattern** used in musical pattern discovery algorithms previously.

Pattern discovery in music in practice, however, needs to take into account the rich temporal, melodic, and temporal-melodic structures in music. One should be very careful when adopting the itemset pattern mining algorithms to the music domain. For example, the algorithms could be inefficient when applied on the note and chord token level, where the notes and chords are treated as separate items without any

relations between them. Nevertheless, itemset pattern mining algorithms may be useful in corpus analysis when more domain specific considerations are made.

(Deep) Machine learning

Many techniques in pattern discovery are strongly associated with machine learning approaches. We already mentioned that this dissertation will be using **ML** techniques such as **Multidimensional Scaling (MDS)**, classifiers, features, synthetic data and so forth. With the right priors from the music domain, many **ML** techniques could be boosted for modelling musical pattern discovery. For example, from a probabilistic point of view, if we categorise unsupervised learning as learning $p(x, y)$, supervised learning as $p(x|y)$, and prediction as $p(y|x)$, we can further crystallise the problem of pattern discovery in these three ways: learning the joint distribution and the marginal distributions between musical events. In fact, there are many works in the probabilistic modelling that are closely related to musical pattern discovery, such as (Collins, 2011; Pearce, 2005).

In the recent rise of many successful **DL** applications, we see some algorithms that could be used for musical pattern discovery (de Reuse & Fujinaga, 2019; Pesek *et al.*, 2017). Generally, there is a cultural excitement about **ML** and **DL**, These algorithms have many strengths, but one weakness is the limited capabilities for generalising to out-of-data distributions. Large **Deep Neural Networks (DNNs)** are known to be universal approximators, which makes the networks powerful, but also makes it possible for networks to simply overfit to remember even completely noisy data via rote memorisation.

Deep learning has not been widely used in musical pattern discovery because training such a system can be costly, and neural networks are data-hungry and hard to control, whereas in musical pattern discovery, there is a lack of annotated datasets, and explainability is often desirable. With small samples and over-training, it is very easy for a large model to simply remember all the data and overfit. Therefore, we do not take this topic as our focus in this dissertation. Nevertheless, there is potential in exploring the methods in this domain, especially using **Recurrent Neural Network (RNN)** models, one shot and manifold learning, and so on (Scerri, 2019). With more and more progress being made on this front, the algorithms following this tradition can benefit from the future advances made in the machine learning community at large.

In any case, it remains true that too complex a system may more easily lead to misalignment of expectations and often imperfect results. Certain forms of "communication channel" between algorithms and humans are more desirable than razor-

sharp accuracy values. Significant progress has been made in this direction of interpretability and explainability, ranging from simple sanity checks to specific methods such as saliency maps. We will discuss more on this topic at the end of this chapter in the Discussion section. We strive to provide more interpretability and explainability in our methods, too.

Time series methods

A time series is a collection of events obtained from sequential measurements over time. In time series analysis, motif extraction aims at finding patterns that form important classes of regularity that exist in a time series. Theories based on time series analysis have been established and applied in areas such as finance and medicine. For example, *Bertens et al.* uses the **Minimum Description Length (MDL)** to extract concise descriptions of discrete multivariate sequential data.

The collision of the term "motif" in music and in time series methods is perhaps not pure coincidence. Both domains face the challenges of finding global and long-term structures in the data. Only with recent inventions, have people begun to attempt to discover motifs longer than 900 datapoints in length (Zhu *et al.*, 2020)¹. Time series methods also have parallels and the potential to be used in musical pattern discovery tasks. Conversely, some time series motif extraction algorithms may benefit from the methods and considerations applied in musical pattern discovery algorithms.

2.4.3 Pattern discovery in other domains

In addition to the musical and the generic algorithms, there are other domain specific pattern discovery algorithms, such as those in bioinformatics, **NLP**, and computer vision. One could draw a parallel between data in other domains with music insofar as an examination of the generalisability of the algorithms and theory. Such a comparison could yield further insights into both the music domain and in other domains. This section will go through a few domains and examine the parallels that have been drawn and could be drawn.

Bio-informatics

The intersection between MIR and bioinformatics has been investigated in (Bountouridis, 2018). A range of insights have been gained regarding music similarity and polyphony reconstruction. For musical pattern discovery, one may also explore making use of algorithms designed for biological motif discovery.

¹for exact motifs was a million datapoints long

NLP

There are significant overlaps between language and music in both the audio and symbolic domain. **NLP** is a self-standing research area, and much ink has been spent on the connections between language and music. Although this is not the focus of this dissertation, there exists too strong a connection not to mention.

We mentioned a few connections in Chapter 2. For example, the syllables can be viewed as notes, prosody as contours, and composition rules as grammar (Bernstein, 1976; Lerdahl & Jackendoff, 1985). Similarly, many text mining approaches could be applied to discovering patterns in music. The research on lyrics is right in-between the fields of **NLP** and **MIR**.

One apparent dissimilarity is that the tokens or patterns in music do not have definitive semantic meanings as in language. Furthermore, the rules for combination and concatenation in music are not as rigid as the syntactic structures in language. The semantic and syntactic flexibility in music introduces extra difficulties such as combinatoric explosion, when it comes to finding patterns and structures in music computationally.

Progress in the field of **NLP** is rapid. We have seen many successful large industrial-level models in real-world applications such as translation tools, conversational scenarios, and creative writing (Brown *et al.*, 2020). What would be the parallels of a wordnet model, a word2vec model, or a GPT model in music? Could they be used to discover musical pattern as well? Recent research has shown promising results that an NLP model can learn meaningful tonal and harmonic relationships in music (Chuan *et al.*, 2020). Musical pattern discovery can potentially benefit from a sophisticated NLP model that can learn to summarise and tokenise music data.

Computer vision

We have used many examples from the vision domain in Chapter 2. One can imagine a parallel between musical pattern discovery and object discovery in images. In fact, computer vision approaches have been used on scores and spectrograms for musical pattern discovery (Calvo-Zaragoza *et al.*, 2018; Costa *et al.*, 2017). By transferring the music representation to the visual domain, it opens up new possibilities for finding the repetition and variation—taking the music data and analyse it as "Ogenmuziek". (Cook, 2000) also commented that

"modern Western notation is a kind of two-dimensional picture of the music, letting you see how it goes at a glance."

Unlike computer vision, though, there is less agreement in music on what defines a pattern or object in music. Moreover, it is more difficult to reach an agreement because listening to music takes more time, even when the stimuli are as short as 3 seconds. A few seconds of time is still much longer than how long it takes to recognise an object visually. The difference becomes more prominent when one must listen for a longer period of time to obtain the context to foreground the patterns. To illustrate the challenge, even more, we can use the concrete example of the cat and dog classification in computer vision pattern recognition. First, there are no categories such as cats and dogs in music, but many ambiguous examples resemble something in-between a cat and a dog. Second, an additional challenge would be that the pixels of the cats and dogs may be gradually presented instead of appearing at the same time.

We can, however, be informed and inspired by many examples in computer vision to improve musical pattern discovery algorithms. Some of the ideas for evaluating algorithms in Section 4.2 were actually inspired by the progress in this field, such as the use of synthetic data and adversarial classifiers.

Other domains

As we mentioned in Chapter 2, many other domains have a connection with pattern discovery. More examples include complex systems science, where complex network models have been used to analyse and generate some patterns music ([ren2014complexity](#); Liu *et al.*, [2010](#)). In the realm of programming languages, a topic we will be looking at starting in the next chapter, methods such as common subexpression elimination may be viewed as a parallel to finding patterns in music.

2.5 Our working definitions

Having looked at all the definitions and peripheral concepts, we can see that it is challenging to define what a pattern is. The terms we introduced are connected with each other and often used interchangeably and imprecisely. Even within the confined context of MIR, the definition of musical pattern varies in the literature, and it is a source of terminological confusion. Like in many cases when defining a common concept, if we try to define once and for all what a pattern is, we can easily run into a cul-de-sac with issues such as context, length, and objectivity and subjectivity. Instead, we attempt to give a practical, flexible, and generalisable definition in the context of this dissertation.

For this dissertation, we propose a working definition of musical pattern consisting of two conditions: first, a potential musical pattern must be a musical excerpt; second, for the excerpt to be recognised as a musical pattern, it must be accompanied by an explanation as to why it is of interest, such as a description of the variation the pattern possesses.

The first part establishes the type of data we are working with, and the second part gives the flexibility and provenance of patterns. The second part also subsumes and can consist of the important concepts we mentioned, including exact repetition and different types of variations. Giving explanations as such is a crucial step towards for algorithms to be perceptually aligned with humans, as shown by the various definitions and concepts we examined. A similar emphasis on the accompanying explanation of algorithmic output can be found in (Colton, 2012), which has seen success in the domain of generative art.

The complexity of patterns then shifts to how to organise the various explanations that will tell us why a music passage is a musical pattern, and how to link the computational methods with human concepts in the explanations. We will make several attempts in later chapters using transformations. However, by the end of this dissertation, we will not have reached the point where we can define numerically and computationally what an adventurous theme (see Section 2.3.2) is, for example.

In the following chapters, our focus is to examine human-annotated and algorithmically extracted patterns. In such instances, we can treat the raw annotations or algorithmically discovered patterns as musical patterns, as the explanation in this case would be the fact that, according to a person or an algorithmic process, they are patterns. We will then build upon these primitive explanations and see if better explanations can be achieved. After Chapter 5, it will become apparent that we can use transformations for this purpose.

2.6 Summary

In this chapter, we examined a variety of pattern-related concepts. By and large, the works reviewed here provide evidence for the generality and importance of the concept of pattern. We hope to have also exhibited that it is almost impossible to define the concept of pattern precisely, comprehensively, and meaningfully owing to the complexity of musical historical events, the many unsolved issues in human perception, social interaction, and the continually changing musical scene.

At the end of this chapter in the last section, we arrived at the working definition of pattern for this dissertation. Our definition is a lenient one, emphasising the pres-

ence of an explanation that accompanies the pattern. This puts us at the starting point of considering which explanations are computationally viable.

With this chapter, we hope to exhibit the complex subject of musical repetition and to address the topic with an array of concepts and terms. As such, we believe that this working definition of musical patterns subsume the many-faceted nature of patterns and may contribute to the long-term enrichment of the area of MIR and other related fields of pattern discovery in general.

In connection to later chapters, we hope that this analysis lays the groundwork for unveiling the complexity behind the concept of pattern. We do not expect all pattern discovery automation efforts to be comprehensive about the concept of pattern and use our definition, but we do encourage them to put some prior thought into this topic in order to establish an informed goal for algorithms.

We mentioned transformations a few times in this chapter. Although we do not discuss the relation between patterns and transformations in-depth here, it will later transpire that we will treat musical transformations as a type of "accompanied explanation" to refine our working definition of pattern. Furthermore, this relation with transformations will also link musical patterns to the use of functional programming, as we mentioned in Chapter 1.

2.7 Discussion

In this chapter, we have offered, albeit partially, a compilation of diverse views from various angles on pattern and pattern discovery. In this section, we open the discussion to several other topics that are, though maybe not critical, are nevertheless intriguing. We use them to expose our limitations as well as extend and consolidate our arguments.

What did we not consider?

There are three points that we omitted from this chapter. Here, we either give a pointer to further discussion on them in the next chapters or explain why we did not include them.

We did not answer the following question: what makes a pattern salient? We pointed to a few broad concepts such as features, repetition, and structure, but we do not have the answer to this question objectively and generally. Without a great number of constraints, saliency is fundamentally subjective. We will touch on this

topic in Chapter 4, where we devise a limited kind of approximation of salience based on the number of patterns discovered at a certain spot in music.

We did not cover the relations between all concepts in Section 2.3, and it would be a Herculean task to cover them all. We introduced 19 concepts in total. As we have reasoned before, if one were to extend the discussion between each one of them, one would have $19 \times 18 = 342$ relations (not considering one-to-many and many-to-one relations) between the concepts to go through, which would not be an easy endeavour. Let alone that this number is not considering the fact that some of the concepts could be defined in different ways by a few different people.

We did not point out explicitly what a pattern is not. Except for the ones that are in contradiction with other definitions, we were unable to locate an entirely "false" definition of pattern. Even the word anti-pattern² is used in other fields to refer to a pattern that should be avoided, but is a pattern nonetheless. This difficulty is not unexpected, however, because giving what is not a pattern renders finding patterns much easier: one can simply define patterns to be whatever that remains from the search space that is not patterns (not considering the subtleties of the negation operation).

Compression

Patterns, repetitions, and ambiguity, as well as a variety of other constructs we had in this chapter, seem to all point to another concept—compression. Compression is a very important concept for the existing musical pattern discovery algorithms and other adjacent fields. In natural languages, it has been shown that ambiguity is needed for the purpose of compressing information (Juba *et al.*, 2011). Once there is a pattern or repetition, it is also often the case that compression comes into the picture (Meredith, 2015).

Although we realise the importance of compression, it has been extensively studied in the context of musical patterns (Meredith, 2013), and it is not the focus of this dissertation. In the following chapters, we do consider compression as a type of application of the discovered patterns.

Where is the hard limit?

As humans, we possess a set of cognitive and perceptual tools to uncover and render explicit the hidden patterns underlying diverse, chaotic, random phenomena.

²another definition of antipattern in musical pattern discovery is the patterns that are infrequent, rare, and under-represented (Conklin, 2013), which is also applicable here.

Is there a strict limit to how far the algorithms can emulate this? At this moment in science, we do not have a precise answer for the question "How is an incoming stream of musical information organised by the ears and brain into percepts, and how do cognitive structures develop with the experience of music?" (Collins, 2011). A direct emulation is therefore not feasible. However, another way to approach this may be to strive for a common language for the alignment between human expectations and algorithmic output. This is precisely what we are attempting to achieve in this dissertation.

Chapter 3 Gathering Human-Annotated Musical Patterns

Data do not exist de novo.

– Dark Data, Hand, 2020

Humans are heterogeneous.

– Human Compatible, Russell, 2019

We mentioned in the previous chapter that data is an important source for pattern discovery. Indeed, the study of the data—musical pattern annotations—has gained increased attention over the past few years. While human annotations are often taken as the reference or training data for developing pattern discovery algorithms, relying on just one reference is often insufficient to capture the complex concept of musical patterns. In addition, there exist varying levels of musical expertise, which may lead to divergent pattern discovery processes and annotator disagreement. In this chapter, we start with a broad overview, which is followed by our efforts in digitising the data from a pen-and-paper annotation experiment conducted in (Nieto & Farboud, 2012b). We explore the potential of digital annotation tools for enabling large-scale annotations of musical patterns, by comparing datasets gathered with two recently developed digital tools, *Pattern Annotation Framework, an annotation tool we devised (PAF)* and *ANnOate MusIC, an annotation tool we devised (ANOMIC)*. We investigate the influences of the tools and different annotator backgrounds on the annotation process by performing inter-annotator agreement analysis and feature-based analysis on the annotated patterns. We discuss implications for further adaptation of annotation tools, and the potential for deriving reference data from such rich annotation datasets for the evaluation of automatic pattern discovery algorithms in the future.

3.1 Overview

Asking human annotators to point out to us what they think of as patterns might sound simple initially. They just need to mark a few boundaries and tell us "those are the patterns" after all. However, the process of annotating patterns is more complex than one might assume. In continuation with some of our discussions in Chapter 2, there are many decisions that a human annotator must make when attempting to annotate patterns in music, such as when a pattern starts and ends, how to find all occurrences of a given pattern in this piece, and how similar must a variation of a pattern be for me to count it as a repetition of the original. During the experiments, annotators may have implicit or explicit strategies or employ conscious or subconscious filters, such as a bias toward the beginning of the music or imposing a minimum or maximum length on their annotations. As a result, many factors are difficult to have perfect control over when setting up a pattern annotation experiment. Even with the same music material, there are factors such as different annotators' musical backgrounds, different functionality between annotation tools, and the exact procedures the annotators followed, to name a few. All of these factors may contribute to different degrees of inter-annotator agreement. In this chapter, we acquire data from multiple annotators and perform analysis to investigate the influences of different musical pattern annotators and annotation tools given the same pieces of music.

MIREX

[MIREX](#) (the Music Information Retrieval Evaluation eXchange, see Chapter 1 for a detailed introduction) contains a collection of tasks, aiming to compare [SotA](#) algorithms and systems relevant for [MIR](#). In MIREX, not only for the task of musical pattern discovery, a significant number of other topics currently researched in [MIR](#) also rely heavily on using reference or "[ground truth](#)", often derived from human annotations. The comparison of [SotA](#) algorithms on the different tasks in the yearly rounds of the [MIREX](#) framework has also uncovered the issue of ambiguity of musical structures for evaluating algorithms. For instance, (Typke *et al.*, 2006) created a new measure for tackling disagreeing perception of similarity. (Flexer & Grill, 2016) discovered a rather low inter-annotator agreement for the MIREX music similarity task, unveiling the problem of using a single reference annotation for evaluating similarity algorithms. (Koops *et al.*, 2019) reached similar conclusions for the MIREX chord estimation task, showing low inter-annotator agreement for chord annotations between musical experts in the Chordify Annotator Subjectivity Dataset. Further-

more, (Balke *et al.*, 2016) concluded that the evaluation of automatic melody finding algorithms is heavily reliant on the human annotator's choice for providing ground truth.

Relevance to pattern discovery algorithms

As discussed in the previous two chapters, the automatic discovery of musical patterns has long been a research topic in computational music analysis (Janssen *et al.*, 2014), and the pursuit of a methodology to compare and evaluate those algorithms has garnered significant community effort, evolving into the MIREX task titled Discovery of Repeated Themes & Sections (Collins, 2011). In this task, the evaluation of newly proposed algorithms is carried out with reference annotations based on music-theoretic analyses by three experts. However, the ambiguity of musical structures and the different conceptualisations of the notion of patterns make the use of reference data relying on only a very small number of experts rather problematic: for the public, there is no clear single comprehensive definition of what constitutes a pattern, or even repetition (see Chapter 1 and 2). Furthermore, not all recurring sequences are perceived as patterns by the listener, as this depends on the structural position of the pattern, the listener's moment-to-moment perception, and other influencing factors such as the listener's musical background or music theoretic education. One can argue that automatically discovered patterns do not have to be perceivable, if they are useful in other contexts, such as for supporting automatic composition (Herremans & Chew, 2017) or classification (Boot *et al.*, 2016). However, in other contexts, it is unquestionably vital that automatically detected patterns are perceivable for listeners, such as in music education (Bamberger, 2000).

More extensive annotation experiments need to be carried out in order to gather reference data for the evaluation of pattern discovery algorithms that rely not only on a very restricted number of musical experts. Gathering annotations from different listeners on the same pieces, for example, could be an initial step in this direction that allows the study of differences and commonalities between listeners regarding their conceptualisation of patterns.

In (Meredith, 2015), it was noted that subjectivity could be reduced by enlisting a larger number of domain experts to reach a more universally agreed-upon ground truth. We would not have a solution for fabricating an inter-subjective **ground truth** by the end of the chapter. Rather than attempting to reduce the differences to some form of inter-subjectivity, our analysis focuses on dissecting the contributory factors of the individual variances. Such a study can pave the way for a more valid eval-

uation of algorithms, based on the consideration of commonalities and differences between listeners or groups of listeners.

The issue of inter-annotator agreement in pattern discovery was previously addressed by (Nieto & Farbood, 2012a) through gathering multiple annotations for a single dataset from twelve annotators on six music excerpts using pen and paper. The data acquired from (Nieto & Farbood, 2012a) faces the problem of reproducibility because they were not digitised. As part of this chapter, we digitise these annotations.

While pen-and-paper annotation has been most commonly used for music-theoretical analysis in the past, the digitisation process afterwards is labour-intensive and error-prone. For carrying out more extensive annotation experiments on musical patterns, digital tools supporting these annotations need to be developed. A digital tool can also provide functionality such as playback, enabling annotators to easily listen to the music.

This chapter

This chapter first introduces the *Human Estimations of Musically Agreeing Notes (HEMAN)* music material used in (Nieto & Farbood, 2012a) and our digitisation process thereof, following which we analyse and compare annotated patterns gathered with two different digital tools, **ANOMIC** (Wells, 2019) and **PAF** (Pesek *et al.*, 2019). The two tools and their experiments were independently developed, but used the same musical excerpts as in HEMAN. While these digital annotation tools overcome the problems of handwritten annotation digitisation, they may influence the annotation process in a different way, such as through different music visualisations. We explore the influence of different annotation interfaces and instructions on the discovered patterns, and study differences and commonalities between patterns discovered by annotators of different musical expertise and from different music theoretic schools. We first explore the gathered annotation datasets by performing inter-annotator agreement analysis, and then employ in a second step: a feature analysis of the patterns by looking into differences between feature distributions of musical patterns in both datasets. This feature analysis allows more detailed insight into the differences between the pattern datasets. Finally, we discuss the implications of the tools and annotators' backgrounds on the annotation process, and the implications regarding future evaluation methods of pattern discovery algorithms based on rich annotation datasets.

3.2 Setup for HEMAN, PAF, and ANOMIC

In this section, we report the music material, the digitisation process, tools, and instructions we use for [HEMAN](#), [ANOMIC](#), and [PAF](#). The data acquisition of [HEMAN](#) is a listening experiment conducted in (Nieto & Farbood, 2012b), using a pen-and-paper approach. For the first time, we digitise these annotated musical patterns. [PAF](#) and [ANOMIC](#) are two musical pattern annotation tools with which two separate experiments were conducted. [PAF](#) and [ANOMIC](#) were both inspired by [HEMAN](#) and use the same music pieces to annotate. We start first by introducing the common music materials for [HEMAN](#), [PAF](#), and [ANOMIC](#). Then, we introduce the tools, experiments, and the differences between them. As a side note, we use these acronyms for three purposes throughout this chapter—naming the experiments, the tools (if used), and the generated annotations.

3.2.1 Music material

Throughout this chapter, we used the same music material. The collection comprises 6 music excerpts as listed below:

- Bach – Cantata BWV 1, Movement 6, Horn (20 bars)
- Bach – Cantata BWV 2, Movement 6, Soprano (15 bars)
- Beethoven – String Quartet, Op. 18, No. 1, Violin I (60 bars)
- Haydn – String Quartet, Op. 74, No. 1, Violin I (30 bars)
- Mozart – String Quartet, K. 155, Violin I (28 bars)
- Mozart – String Quartet, K. 458, Violin I (54 bars)

These pieces have been selected by (Nieto & Farbood, 2012b) for pattern annotation experiments due to their different musical characteristics. For example, the Bach chorale is short and has very little rhythmic variation, while the Beethoven string quartet is long, and has clear repetitions varied in pitch. In the original paper (Nieto & Farbood, 2012b), more detail was given:

"Excerpts 1, 5 and 6 proved to be particularly difficult to parse for humans, ... The data resulting from these 'difficult' excerpts enable us to ascertain the amount and type of overlap that frequently occurs in motive perception. Excerpt 3 (shown in Figure 1), on the other hand, has more clearly defined motives that are readily apparent from a quick glance at the score."

For long pieces, we use roughly the first page of sheet music (precise bar numbers are included in the listing).

3.2.2 HEMAN

One of our contribution in this dissertation is that we digitised the annotations in (Nieto & Farbood, 2012b) and open-sourced the annotations¹. Figure 3.1 shows an example of the annotations before digitisation. This section introduces the instructions given, the subjects of this experiment, and the digitisation process.

Instruction

The annotators were provided with audio excerpts. The audio excerpts were MIDI files that were synthesised with changes in dynamics based on the notation on the scores, but without rhythmic variations such as fermatas. This simplification eliminates the effects of timbre and choices made by performers, which might influence or distract the subjects. The fermatas in the second piece by Bach are arguably structurally significant, and we left them out to see if the melody alone is sufficient for the annotators to detect these structures.

More specifically, the following instructions were given to the annotators:

"Please, analyze the following musical excerpts and mark all the musical motives you can find. A musical motive is defined as a short musical idea, a salient recurring figure, musical fragment, or succession of notes that has some special importance in or is characteristic of a composition. It shouldn't be longer than a musical phrase. If you find a motive that is similar to another (or multiple versions of a motive), choose the one that you think is the most representative. Even though all motives are relevant, please rate each one of them from 1 to 3:

- 1 = Not as relevant
- 2 = Relevant
- 3 = Highly relevant

You can listen to the music excerpts as many times as you like. You can find them here. <http://urinieto.com/NYU/Research/MotivesExperiment/>"

We deliberately offered several possible interpretations as to what defines a "musical motif" and allow the subjects to both analyse the pieces and use their musical intuition on what constitutes a musical motif in the process. As Henri Poincaré puts it, "It is by logic that we prove, but by intuition that we discover."

We did not ask them to laboriously label all occurrences of the same pattern for us. This will turn out to be a critical difference between HEMAN, ANOMIC, and

¹<https://github.com/irisypingren/HEMANanalysis>

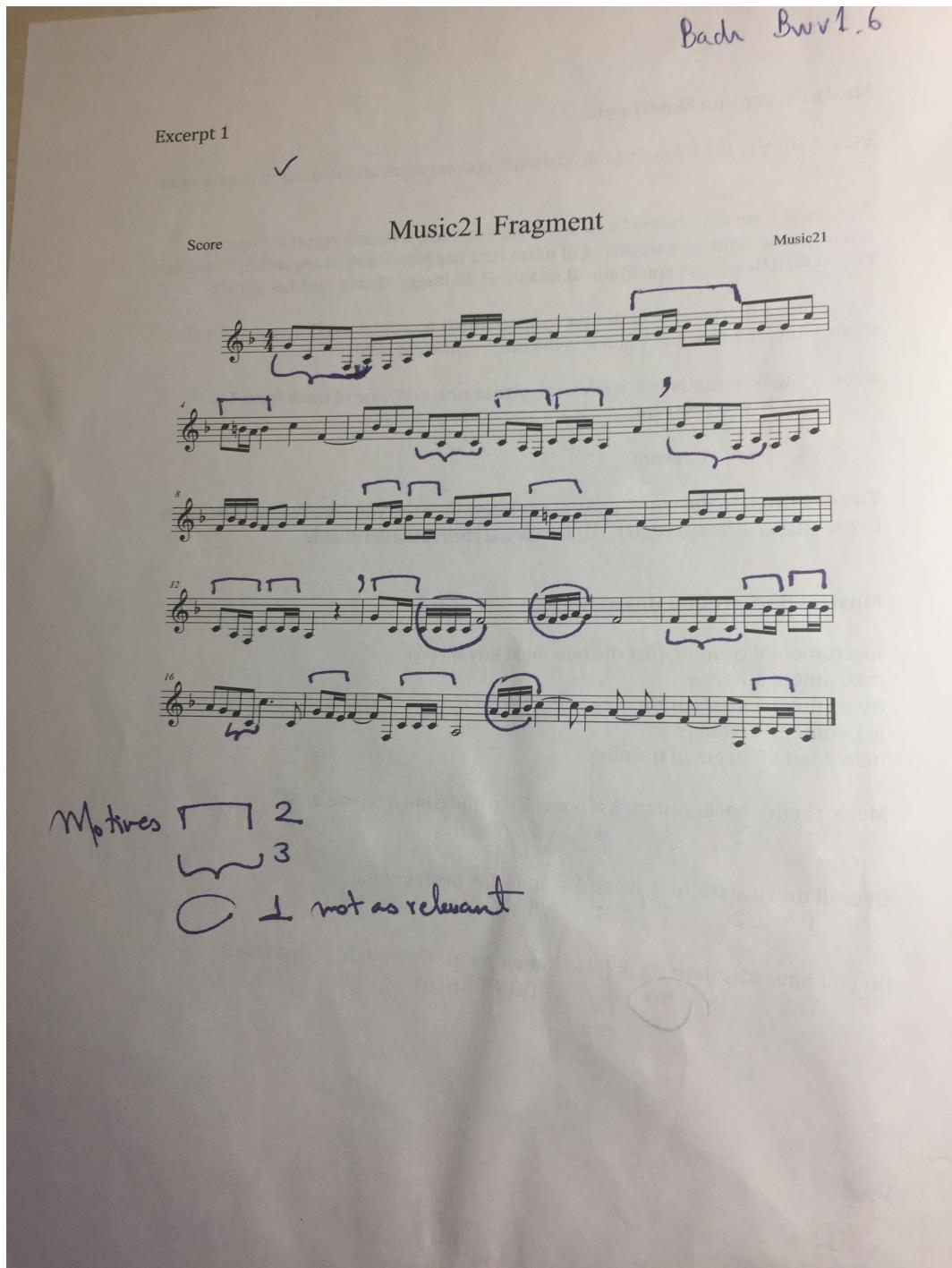


Figure 3.1: An example of the raw paper-and-pen annotation.

3 Gathering Human-Annotated Musical Patterns

PAF. Without manually annotating pattern occurrences, we lose critical information about the repetitions of these patterns, but we obtain at least the prototypes of musical patterns the subjects perceived in the piece. We try to remedy and improve on this aspect in PAF and ANOMIC, respectively.

In summary, the annotators were exposed to the sheet music and audio, and were asked to identify music motifs and patterns which are shorter than a musical phrase. Subjects also labelled the degree of relevance or confidence value of each pattern as Low(1)/Medium(2)/High(3). Each pattern is thus represented by a set of pitch and duration pairs plus its associated relevance/confidence label. Unlimited time was given to the subjects, and we did not reveal the names of the pieces on the annotation sheet.

Subjects

We asked 12 subjects to annotate patterns in six musical excerpts. The annotation process for obtaining the dataset was conducted at New York University (NYU). The subjects were all graduate students at NYU and had an average of 10 years of formal musical training (Standard Deviation = 2.3). In Table 3.1, we report the results of the musical background questionnaire of our subjects. These 12 annotators produced a total of 715 annotations.

Inst	Year	Inst2	Year2	Theory	Overall	Abs
Piano	10	Violin	4	3	3	0
Oboe	5	Piano	1	2	3	0
Piano	5	None	0	3	3	0
Piano	12	None	0	3	3	0
Piano	14	Trumpet	8	4	4	0
Guitar	13	Bass	0	5	5	0
Piano	13	Violin	4	4	4	0
None	0	None	0	0	0	0
Guitar	8	Piano	4	5	5	0
Guitar	10	Kazoo	4	5	5	0
Violin	8	Guitar	2	3	3	0

Table 3.1: Musical background of annotators. The columns Inst and Inst2 denote the main and the secondary instrument the participant plays, followed by how many years of experience. The headers Theory and Overall denote the self-rated musical theory background level and the self-rated musical background level. A professional background has a score of 5, and no background has a score of 1. The header Abs stands for absolute pitch/perfect pitch, with 0 stands for self-identified as no perfect pitch, 1 otherwise.

Digitisation

There are advantages and disadvantages to using a paper-based setting. On the one hand, this could preserve the most natural mindset on perceiving patterns in music; on the other hand, it may present some risk of incorrectly interpreting or digitising the markings. For example, in Figure 3.1, it is not immediately apparent whether to include the quaver of A in bar 3 into the musical pattern. The same situation applies to the crochet in bar 14 in the annotated pattern. The rule we followed here was to take the midpoint of the gap between the two printed notes in question and include/exclude the note in the patterns based on which side the annotation started/ended with respect to the midpoint. For example, in both bar 3 and bar 14, we include the quaver and the crotchet. Further, since the annotators were not forbidden to mark the occurrences in addition to the prototype patterns, we do find such occurrences in the photo. We take those occurrences into account as long as a relevance score is given.

To digitise this notation, we first calculate the time intervals [start time, end time] of the pattern annotations. We take the unit of crochet in this time interval format. For example, the first annotated pattern in the time interval format is [0,2], and the second pattern is [8,10]. Although transcribing from the photo format to the time interval format is relatively time-consuming and prone to human error, we do not know of any mature technology that could automatically convert between the two formats. Although optical recognition techniques are promising, they may produce low accuracies with imprecise markings. A digital annotation system would be ideal for circumventing this problem. For conducting a large scale online experiment, it would only be viable by developing such an annotation system.

3.2.3 PAF

The [PAF](#) tool² was developed as an online application (Pesek *et al.*, 2019), with the Django framework serving as the back-end, and with the MIDI Player and Verovio JavaScript libraries serving as the front-end visual representation of the music sheet. The tool shows the music score of a selected piece of music, as seen in Figure 3.2. Thus, it is designed to be used mainly by individuals with a certain degree of musical expertise, namely, those who can at least read staff notation. The user can annotate music patterns by clicking on the beginning and the ending note of the desired range in the score. The user can also set the name and the relevance (1–3) of an individual annotated pattern. To aid the annotator, the displayed score can be listened to with

²Tool available at <http://framework.musiclab.si>

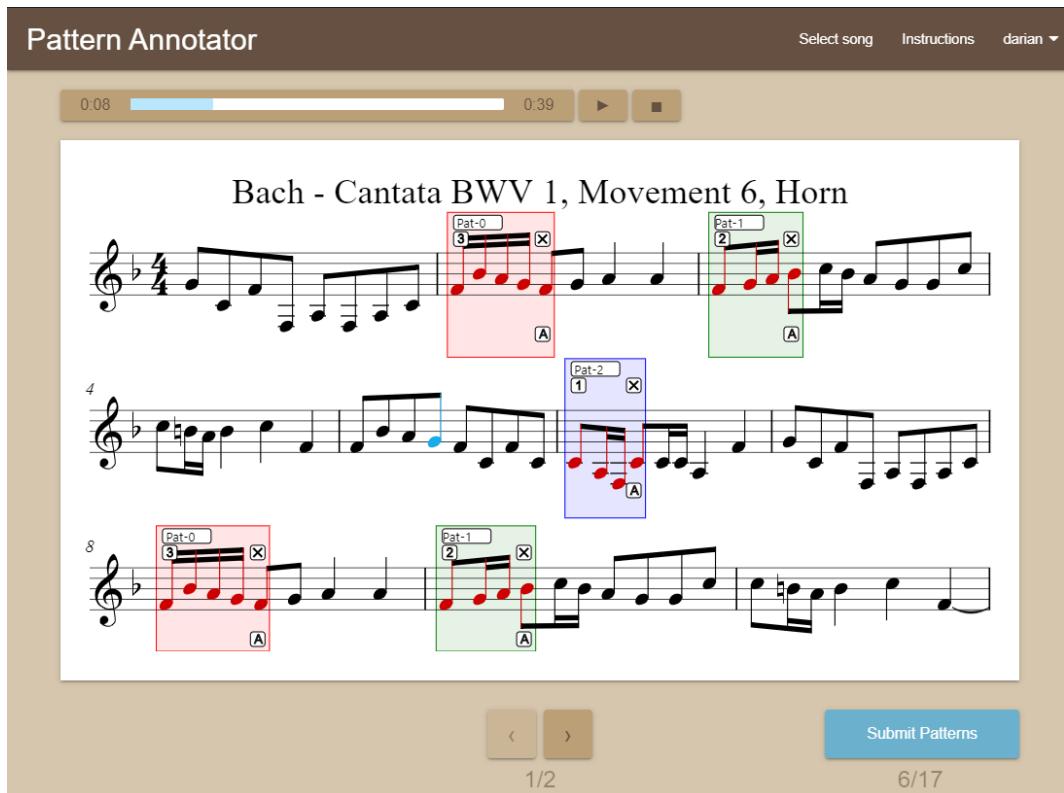


Figure 3.2: An example of an annotation session in PAF.

our synthesised MIDI files. While playing the score, the notes nearest to the current time in the music playback are highlighted. The tool also contains a feature for automatic annotation, which annotates other occurrences of the selected pattern. While developing the framework, a major focus was on the responsiveness of the layout to accommodate different screen sizes and devices (mobile, desktop). In the case of smaller display size, the score is split into multiple sheets. By open-sourcing the tool³, we hope to aid other researchers in the MIR field who are dealing with pattern-related data gathering and invite them to contribute additional features.

Instruction

The following instructions were given to the PAF annotators:

The goal of the site is to allow users to find and submit patterns found in music.

A musical pattern or motive is defined as a short musical idea, a salient recurring figure, musical fragment, or succession of notes that has some special importance in or is characteristic of a composition. It shouldn't be longer than a musical phrase. If you find a motive that is similar to

³Source code available at <https://bitbucket.org/ul-fri-lgm/patternannotationframework>

another (or multiple versions of a motive), choose the one that you think is the most representative.

To begin, click on a note to mark it as the start of a pattern. The starting note will be coloured blue. Next, click on another note to mark the end of that pattern. All of the notes between the start and the end will be coloured red, thus representing a marked pattern. To mark multiple patterns simply repeat the process.

Each pattern also includes an ID field (top left corner), which can be used to change the ID of a pattern. To denote multiple pattern occurrences, the ID of a repeated pattern can be changed to match the ID of the original pattern. Alternatively, exact repetitions of a pattern can also be annotated automatically. By pressing the A button on a pattern (bottom right corner) all repetitions of the selected pattern will be automatically marked. Once a pattern is finalised you can also rate it from 1 (not as relevant) to 3 (highly relevant).

To delete an UNFINISHED pattern click on its starting note, which is coloured blue.

To delete an already FINISHED pattern press on the X button, which can be found in the right corner of each pattern.

To navigate the sheet music use the buttons located below the sheet music. You can also use the arrow keys on your keyboard to achieve the same result. Once you are finished click on the SUBMIT PATTERNS button and click OK. The patterns will then be submitted and the site will take you to your next task.

To zoom out or in press - or +.

To use the music player simply click on the play button. You can also navigate the song by clicking on the song progress bar in the top left.

The description of a musical pattern is the same as HEMAN. A range of descriptions of functionality was added to enable the user interaction with music and patterns.

Subjects

The PAF tool was used by 13 students attending three university-level music study programmes: 4 students from the musicology masters programme at the Faculty of Arts, the University of Ljubljana (in the following termed as MU), 3 students of the Music Academy, including music theory and composition at the University of

3 Gathering Human-Annotated Musical Patterns

Ljubljana (termed as TC), and 6 students from the music pedagogy program at the Faculty of Education, the University of Maribor (termed as PE). The PAF website included a description of the tool and a summary of the tool's features. The majority of participants were students aged between 20–25 having between 5–10 years of instrument experience and between 8–15 years of music theory experience. In total, 373 annotations were gathered from 13 annotators over the course of a month.

3.2.4 ANOMIC

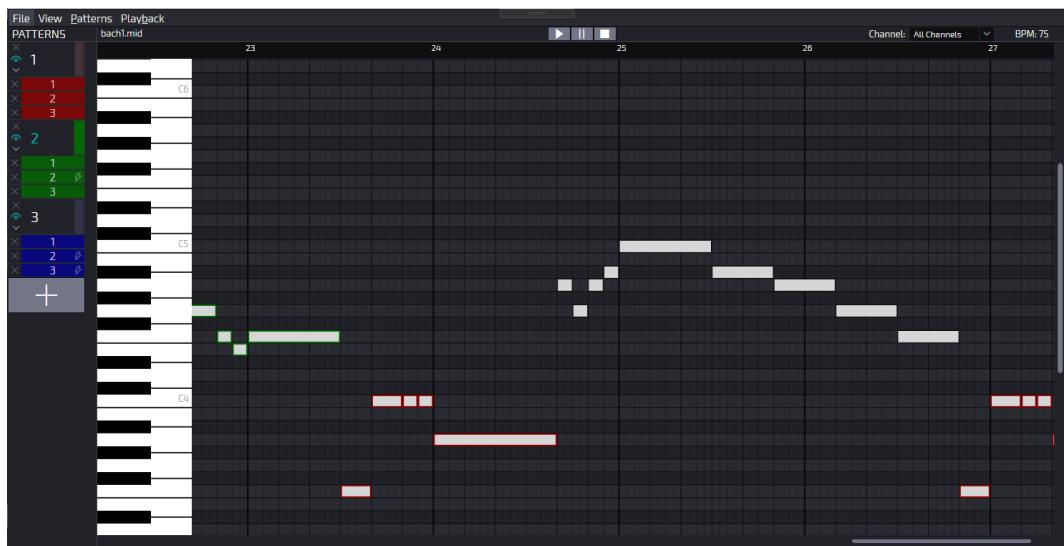


Figure 3.3: An example of an annotation session in ANOMIC.

The ANOMIC pattern annotation tool was created as a stand alone application for the Microsoft Windows operating system (Wells, 2019). The main view of the tool visualises the MIDI representation of a music piece as a piano roll by plotting the musical notes as rectangles on a two-dimensional onset-time-pitch canvas. This approach is also commonly used in music editing software, as it allows for easy interaction with MIDI elements (most commonly musical notes). Figure 3.3 shows ANOMIC's interface, where the piano roll representation is visible.

For the experiment of ANOMIC, a demo video for the experiment was provided for making instructions on the pattern annotation accessible to users even with little musical training. An instruction document was also given in written form. Below, we include the most relevant instructions in words.

Instruction

With ANOMIC, a group of users installed the tool on their computers and were then asked to submit their annotations via email, along with a completed survey about

their background. If we take away the instructions for installing the software, the more relevant instructions for the annotation experiment are:

... Annotate repeated patterns in the music. Patterns are distinct, short musical segments or phrases that are considered to be characteristic to a given piece of music and appear multiple times throughout the piece. Be sure to listen to the music and annotate these patterns and their occurrences using the tool. The occurrences don't need to be exact matches, but they should be closely related (compare this to finding occurrences of a leitmotif in a film soundtrack, for example).

For any occurrences where you're not sure whether or not they belong to a pattern, right-click on them to set their confidence level to "2" (unsure) or "1" (highly uncertain).

Automatic Occurrence Matching At any point during the session, the user can right-click on an occurrence that they had already found and initiate the automatic occurrence matcher for that occurrence. It will comb through the entire track and find occurrences that are exact repetitions of the occurrence in question, even if these repetitions may not be in the same time or pitch position. This implies that it is transposition-invariant and also works through polyphony.

We see again that the description of a musical pattern is similar to HEMAN and ANOMIC. A new functionality was added for finding exact and chromatically transposed occurrences of annotations. We will be looking at a cross-comparison between the tools and the experiments in the following sections (Section 3.2.5 and Section 3.2.6).

Subjects

The ANOMIC tool was used in an annotation experiment with 26 participants of varying musical backgrounds and demographics⁴. In total, 788 patterns and 2763 occurrences were gathered, annotated by 26 participants with diverse backgrounds. The participants' levels of musical expertise were assessed through a survey including 10 questions (e.g. ability to read sheet music, proficiency in playing an instrument, academic degree in music), leading to a score between 1 and 10. As a rough approximation of their musical expertise, the annotators were divided into two groups with a cutoff of 5, dubbed the musicians (14 participants) and the non-musicians (12 participants) (see (Wells, 2019) for all details).

⁴Available at <https://github.com/StephanWells/ANOMIC-dataset>

3.2.5 Differences between the tools

We reviewed two digital pattern annotation tools that were developed concurrently—ANOMIC and PAF—and two annotation experiments conducted separately using these tools. While both tools allow for more accessible digital annotation of patterns, they differ in their implementation, functionality, interface, and music representation. The most glaring concerns the different visualisation of music they employ: piano roll or sheet music. While these visualisations of music may seem equivalent at first glance, they represent the same information in different ways that can in turn influence the users.

For example, the music sheet of the PAF tool can show a larger part of a music piece than the ANOMIC tool, in which the notes can get too small when zoomed out. The visual presentation of note durations is also different. While notes are of similar sizes on the music sheet, independent of their duration, the notes on the piano roll differ tremendously in their size. Therefore, longer notes might thus attract more attention and be harder to scroll past. On the other hand, the sheet music representation can also present a better user experience on smaller screens. The music sheet can also contain additional information, apart from the actual notes, such as auxiliary signs for timbre and volume, which can influence the users' understanding of the music piece.

The tools also differ in how they enable the selection of patterns and the annotation of their occurrences. The controls are smoother in the ANOMIC tool, which provides the ability to use click-and-drag actions to select patterns. Conversely, annotating in the PAF tool is done by clicking on the starting and then ending note of a pattern. The click-and-drag approach could be perceived as more intuitive, especially considering the left-to-right piano roll visualisation, whereas demarcating the beginning and ending of a pattern makes the annotation of longer patterns faster. When the annotation starts, for both tools, a default pattern number (ID) is given at first, and if the users proceed to a different group of pattern occurrences, they can use a new number to signify this new group.

ANOMIC also has a helper function to enable users to automatically annotate exact repetitions and chromatic transpositions of already annotated patterns, as implemented by an automatic occurrence matching function (Wells, 2019). This function can ease the labour-intensive search necessary in order to annotate all occurrences of a pattern, which is non-trivial for annotators (see (Volk & Van Kranenburg, 2012)). Although it was not mandatory to use this function during annotation, in a follow-up survey, we saw that the helper function was used at least 9 times by all participants, and that the function was liked by some annotators but one participant said that

it "felt like cheating". A detailed user study of ANOMIC and the helper function in particular can be found in (Wells, 2019). While PAF was later updated to include such an automatic annotation feature, it was not available in the version used for gathering the data for the current analysis. Regarding other basic annotation controls, such as pattern deletion and tagging, the tools have similar functionalities.

The tools take different data gathering approaches. ANOMIC is a standalone tool and can be used offline. The gathered patterns must then be submitted via external means, which can affect the number of gathered submissions. Meanwhile, PAF is implemented as an online tool, which automatically saves the annotated patterns in a database. In addition, users must first register and provide their background information to access the tool. In comparison, the ANOMIC tool does not come with built-in registration or user information gathering functionality, so researchers must resort to online surveys. The tools also differ in the overall annotation process: PAF does not allow re-annotation of music pieces and only allows annotation of the provided music pieces, while ANOMIC allows annotation of any music piece the user provides.

Both tools allow for the playback of the presented music piece. When music is played in PAF, the currently played notes are highlighted, and the sheet pages change when the last note on the page has been played. The tool also has a separate music player with which the user can navigate through the song. The ANOMIC tool takes a different approach by providing a tracker, in the form of a vertical line that denotes the current position of the music.

These differences and similarities are still subject to change. Like many technologies nowadays, annotation tools are evolving. Future updates will most likely focus on the addition of user-friendly features, and in particular, how to enable the users to more easily modify the tools (e.g. adding helper functions) while annotating. We summarise the main differences between these two tools in Table 3.2.

3.2.6 Differences between the experiments

The description of a musical pattern is similar across all experiments. For the PAF tool, the instructions for the participants were actually designed to closely follow those used in (Nieto & Farbood, 2012b). For ANOMIC, the intentions were to improve upon the instructions of (Nieto & Farbood, 2012b) in conjunction with the helper functions.

In summary, there are two marked differences between the instructions: whether participants were asked to listen to the music, and how to annotate occurrences of a pattern. While the instructions for using ANOMIC explicitly mention the impor-

Tool	ANOMIC	PAF
Platform	Windows	Online
Visualisation	Piano roll	Sheet music
Background information gathering	External survey	Online registration
Musical background	Self-rated music skills	Music pedagogy and theory students
Helper functions	Auto-tagging exact rep. and chromatic transp.	None

Table 3.2: Summary of the main differences between the ANOMIC and PAF annotation tools and their experimental subjects. "rep." stands for repetition, and "transp." stands for transposition.

tance of occurrences, PAF users and HEMAN participants are instructed to choose a representative pattern when they see multiple similar ones. In the experiment using ANOMIC, the participants were asked to listen to the music, whereas with PAF and HEMAN, it was not explicitly asked.

In terms of the differences in instructions regarding whether the annotators listened to the music, we could not check whether the participants actually listened to the music for either experiment. This can be accomplished in future work by logging annotators' behaviours and interaction with the annotation tool, so that we see when and for how long the annotators have been listening to the music.

In terms of the differences in instructions regarding whether annotators annotated all occurrences, it is a more complex problem. We believe that annotating all occurrences can be more helpful for developing musical pattern discovery algorithms. However, without some helper functions, such as the ones in ANOMIC, humans might not be able to or lose patience in meticulously annotating occurrences of a pattern without errors. For PAF, although we could not conduct the experiment with the helper function, we used the newly added helper functions post-experiment to augment the collected data by identifying their repetitions. We will see that this does not entirely equalise the PAF and ANOMIC experiments, but at least we add the notion of pattern occurrences in the way we can. As a result, for subsequent comparisons, we make comparisons only between ANOMIC and PAF data where the notion of pattern occurrences is present to an extent.

These discrepancies are not ideal for perfectly controlled comparisons. However, since these tools and experiments were developed separately with the same goal—gathering data for musical pattern discovery, there are values and lessons we can learn from looking at the results in parallel. We leave a more controlled experiment for future work and examine what comes when diversity is allowed during the setup of a musical pattern data collection experiment.

3.3 Our methodology and other metrics

Our analysis has three exploratory aspects: differences between annotators with different backgrounds, differences between annotating the most representative pattern and all occurrences of a pattern, and differences between annotations gathered with two different tools. All aspects will be explored by analysing inter-annotator agreement and by analysing the distributions of various pattern features, to gain deeper insights as to why annotators might have disagreed. In the following, we describe these methods in more detail.

3.3.1 Agreement analysis

To measure inter-annotator agreement, an important concept is that of "matched" annotations. Given two pattern occurrences P_1 and P_2 , with beginnings and endings denoted as b_1, b_2 and e_1, e_2 , were considered to be matched when $|b_1 - b_2| + |e_1 - e_2| \leq T$, where T denotes a threshold value. The vertical bar notation indicates "taking absolute value" to disambiguate from taking cardinality of sets.

Given two sets of pattern occurrences R and C from two annotators, we call one set the reference (R). It does not matter which set is taken as the reference, because we will eventually consider the other set as the reference as well. Using $\#$ as "the number of" sign, we then calculate the commonly used measures, namely, *precision*, *recall*, and F_1 score for all possible pairings of annotators, as specified in the equations below. Each individual annotator is compared to every other annotator.

$$\text{Precision} = \frac{\#\text{matched annotations}}{\#R}$$

$$\text{Recall} = \frac{\#\text{matched annotations}}{\#C}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

With a number of annotations in a piece, we can expect the precision, recall, and F_1 score to give us a summary of the agreed patterns between any pairs of annotators. These measures will fit the intuition that the more far apart the different annotated pattern boundaries are, the more they disagree; the greater the number of patterns the annotators disagreed upon, the more two annotators disagree. For example, if annotator A noted that the second bar of a musical piece is a pattern, while annotator B included the last quaver of the first bar and the second bar as a pattern, we have the same pattern ending, but a slightly different beginning. The threshold value gives us the flexibility to configure whether the two annotators should be considered to be in agreement (matched) or not. In the example above, if the two patterns are the only annotations in the piece, we have an F_1 score of 0 if $T < 1$ quaver, 1 otherwise. In this way, we can see how much disagreement there is on different scales of time resolution. The reason that we focus on the beginnings and endings of patterns is that, within the same piece of music, once the beginnings and endings are determined, the excerpt's content is the same for monophonic melodies.

In the following analysis, the starts and ends were measured in crotchets, and the threshold was set to 5 crotchets as the default for this chapter, following (Ren, Koops, *et al.*, 2018). We will also see that a threshold of 1 crotchet was used for comparison

later on. In future work, other threshold values or dynamic thresholds should be investigated.

In relation to MIREX's metrics

This measure does not have direct correspondence with the occurrence or establishment scores in MIREX. MIREX measures are based on the cardinality score, which considers the actual musical events themselves (pitch and duration), and not their position in the piece (see next chapter Section 4.2.1 for more detail on MIREX measures). There is an indirect connection when considering that the musical events are the same given the same position in the same piece. However, the "matched annotation" for a single occurrence is a binary notion in our case, while in MIREX measures, each occurrence has a fractional value in relation to another occurrence, and then a maximum is taken. In summary, the computational process and the focus of the evaluation are both quite different between our measure and MIREX's.

In relation to other inter-rater reliability measures

Other inter-rater reliability and agreement measures exist in literature. However, it is not straightforward to convert the annotations of musical patterns to be compatible with other inter-rater reliability measures, such as Krippendorff's alpha coefficient. To calculate these measures, one usually starts with a matrix of $n \times m$, where n is the number of annotators, and m is the number of attribute values that will get annotated. In our case, we do not have this matrix, because each annotator might annotate a different number of patterns that need to be "matched" first, as we devised in our measure. What follows from this matching step is then counting the number of "matched" annotations and calculating corresponding percentages. As a consequence, our measures are intuitive and easy-to-understand. We do agree that our measures are simplistic and can be improved with more sophisticated statistical methods in the future.

3.3.2 Feature analysis

In order to further explore the differences between annotations of the two digital tools, we compared both annotation datasets on 33 different pattern features, of which we report 7 here, for simplicity, while all 33 musical features are described in detail in supplemental online material⁵. The majority of the features were inspired by the work of (Collins, 2011), in which musical patterns were rated based on

⁵Results available at https://bitbucket.org/anonymous_submitter/agreement-in-musical-patterns

3 Gathering Human-Annotated Musical Patterns

a plethora of musical features from previous research, including (Cambouropoulos, 2006; Conklin & Bergeron, 2008; Forth & Wiggins, 2009; Meredith *et al.*, 2002; Pearce & Wiggins, 2007). Several features were also inspired by the research of (van Kranenburg *et al.*, 2013), which compared global and local features of folk song melodies.

To compare features of the patterns, we take the first occurrence of each pattern annotated in ANOMIC, as PAF annotators only annotated the most representative occurrence of each pattern. We take the first occurrence because the first occurrence of a pattern in a musical piece tends to have a more significant role according to (Schoenberg, 1967). This taking-the-first-occurrence approach has an exception for one feature that we will introduce later, where we use all occurrences from ANOMIC to see whether the annotators actually followed the instructions closely regarding annotating single or multiple occurrences.

We then compare features of the patterns, we computed musical features of each pattern, thus forming feature distributions for both datasets. Next, the distributions of each feature were normalised by taking the minimum and maximum values across both distributions and performing min-max feature scaling. Thus, we obtained feature distributions with a range of 0 to 1, which are visualised in Figure 3.5. The computation process of each analysed feature is described in detail in the supplemental online material⁵, which also includes the Python source code used for the analysis.

In Table 3.3, we list the 7 features included in this paper, namely those which we consider to be most intuitively related to pattern characteristics perceivable by users (such as the duration of the last note or the note range). Notice that the Occurrence feature is what we mentioned as an exception to the taking-the-first-occurrence approach above. Furthermore, following the comparison of local and global features in (van Kranenburg *et al.*, 2013), we focused on local features, which are more likely to be assessed by humans when annotating patterns. We further reduced the number of important features by analysing the Spearman's and Pearson correlation coefficients between pairs of features. The highest Pearson correlation value appeared between the feature pattern duration and note range, which has a correlation of 0.64 and 0.65 for ANOMIC and PAF. Tables of the most correlated features for each dataset can be found in the supplemental online material⁵.

Our last step to analyse pattern features is based on independent two-sample Student tests (t-tests), two-sample Kolmogorov-Smirnov tests (KS-tests) and boxplot visualisations of distributions. Given our null hypothesis that the samples are drawn from the same distribution, and that we are unsure whether our data is normally distributed, we use both parametric and nonparametric tests to verify how likely it

Name of feature	Description
Pattern Duration Occurrences	The duration of a pattern in crotchets The number of times that a pattern occurs
Last Note Duration	The duration of the last note of a pattern
Note Range	The number of semi-tones between the lowest and highest note of a pattern
Pitch Direction Changes	The number of melodic arcs in a pattern
Intervallic Leaps	The fraction of all intervals of a pattern that are intervallic leaps (> two semitones)
Root Notes	The fraction of notes in a pattern that are root notes or their octaves.

Table 3.3: Feature descriptions

is that the distributions actually differ. The differences are represented by high t and D statistic values in combination with low p values of the performed tests. We also considered boxplot visualisations of the distributions to better understand the shapes of the distributions and the differences between them.

3.4 Results

In this section, we report the results of comparisons between PAF and ANOMIC. We do not compare them together with the HEMAN data as explained in Section 3.2.6: the annotated patterns were not grouped into a pattern-occurrence relation, and therefore difficult to compare with those that were grouped. We use the two kinds of method introduced in Section 3.3: agreement analysis and feature analysis.

3.4.1 Agreement analysis

In order to analyse inter-annotator agreement, we computed the F_1 scores of all annotator pairs across all music pieces. We gathered these values into F_1 score matrices, which allow us to visually inspect the results in a compact manner. Figure 3.4 shows these matrices, in which annotators are grouped based on the annotation tool used and their backgrounds. The analysed groups include the TC, MU and PE groups of PAF as well as the musician and the non-musician groups of ANOMIC (see Section 3.2.3 and 3.2.4 for the setup of the experiments). Based on the obtained inter-annotator agreement values, we refer to the values above 0.95 (yellow matrix values) as indicators for a strong agreement in this paper. It should be noted that the number of annotations was not split equally among music excerpts, as some annotators of the PAF tool did not annotate the last three excerpts. This was likely due to the fixed order of music excerpts and the significant time investment in the annotation process. Once this issue of the PAF tool was identified, it was reported and addressed by the developers, who randomised the ordering to improve the tool for future use.

We will not make a comprehensive cross-comparison between ANOMIC and PAF using our measure introduced in 3.3.1, because of the single- or multiple-occurrence difference in the instructions of the tools. The concept of "matching annotations" is a complicated one if we compare the most representative occurrence annotation of a pattern with all the occurrences annotated for a pattern. In addition, "the most representative" and "all occurrences" are not guaranteed as the annotators can only do the best they can. We will, therefore, leave this to be explored in future work.

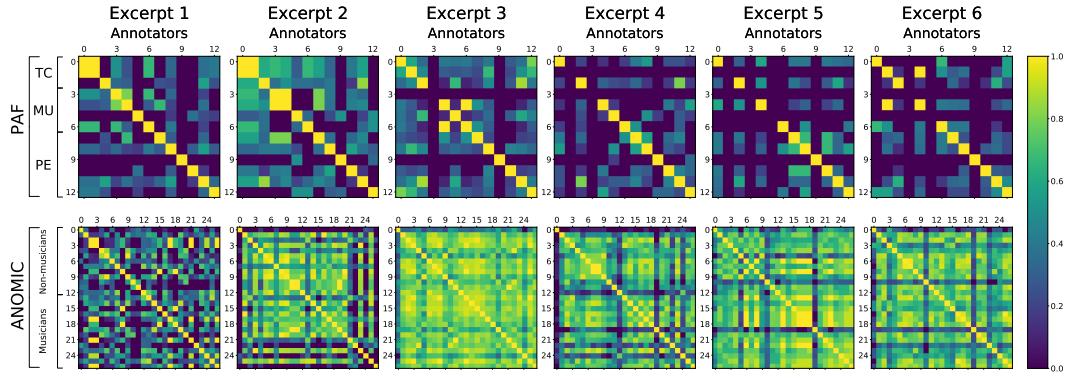


Figure 3.4: F_1 score matrices, representing the pairwise agreement between annotators, with the time threshold set to 5 crotchetts. Each matrix showcases results for a different music excerpt. Matrix columns and rows denote different annotators, grouped based on the annotation tool used (PAF or ANOMIC) and their musical background. Matrix cell colours correspond to the obtained pairwise agreement values, where yellow denotes high agreement and blue indicates low. Some annotators did not provide annotations for all excerpts, which can be seen along the diagonals as low agreement.

PAF

The TC and MU groups show strong agreement on the first three music excerpts. Meanwhile, the results of the PE group show many weaker agreements, despite a more significant number of annotators. Contrary to the TC and MU groups, the annotators of the PE group did somewhat agree with the annotations for excerpt 4. There is only one strong agreement between annotators belonging to different groups (excerpt 6, annotators 1 and 4). Since there are several strong agreements between the annotators within individual groups, the lack of agreement between different groups could indicate the potential influence of different study programmes on the annotators' perception of the most representative musical patterns.

ANOMIC

For ANOMIC, the agreement values for the two subgroups are similar: the average inter-annotator agreement for the musician group is 0.61, and 0.63 for non-musicians. While some agreement does exist between the ANOMIC and PAF annotators, we only observe one strong agreement between two PAF annotators and one non-musician annotator of ANOMIC in excerpt 1. We do not make further comparisons between PAF and ANOMIC disagreements, as a range of factors could have contributed to their differences, such as differences in instructions and the threshold value.

A different threshold value

Next, we lower the threshold for agreement computation from 5 crotchets to 1 to see what changes may be brought on to our results by a different threshold value. The agreement values become much smaller among the ANOMIC non-musicians (0.38) and the musicians (0.47), for the crotchet threshold of 1. For the same threshold the average agreement between all ANOMIC participants was 0.40.

A note on taking average

We have an additional note for these comparisons. In taking averages, we can compare between groups while marginalising the effects of individual differences between musical pieces. We are aware that this is not always valid because there is a varying degree of difficulty in finding patterns across different musical pieces. It is possible that a group of annotators disagree strongly on one single piece and agree perfectly with each other on the rest, which would be obscured in the average, with the music being a confounding factor. However, in Figure 3.4, we see a range of disagreement and agreement. Admittedly, excerpt 1 is more disagreed upon than others, so we also calculated the values by only using the other five excerpts, and the results and conclusions did not change. Furthermore, the computation of the average was based on the whole matrices, thus including values where users did not provide any annotations. These values were simply set to 0 and were included in the computation. We also analysed the average values if these values were ignored. Despite affecting the average values of the comparison, the changes in values were not significant since the values, based on PAF users, simply increased by around 0.01. Thus, we decided to only report the original values, which included missing annotations.

3.4.2 Feature analysis

As introduced in Section 3.3.2, we analysed, for each pattern feature, the annotations of ANOMIC annotators with musical and non-musical backgrounds and compared them to the PAF dataset, whose annotators all had a musical background. We investigated whether the difference between datasets was also present in these background subgroups to identify if the observed difference between the PAF and the ANOMIC dataset was influenced primarily by the tools or the musical backgrounds of the annotators.

The analysis results revealed 23 features, where significant differences were seen between the annotations of the PAF and the ANOMIC tool. We then eliminated sev-

t-tests between pattern features (t and p values of t-test)		PAF / ANOMIC	PAF / Mus.	PAF / Non-Mus.	Mus. / Non-Mus.
Pattern Duration	7.54 (9.60 $\times 10^{-14}$)	4.06 (5.38 $\times 10^{-05}$)	7.68 (5.09 $\times 10^{-14}$)	4.91 (1.09 $\times 10^{-06}$)	
Occurrences	-11.96 (4.35 $\times 10^{-31}$)	-16.42 (2.98 $\times 10^{-51}$)	-11.12 (1.13 $\times 10^{-26}$)	-3.1 (1.98 $\times 10^{-03}$)	
Last Note Duration	6.75 (2.45 $\times 10^{-11}$)	3.93 (9.30 $\times 10^{-05}$)	7.36 (4.90 $\times 10^{-13}$)	3.49 (5.15 $\times 10^{-04}$)	
Note Range	5.18 (2.57 $\times 10^{-07}$)	1.62 (1.05 $\times 10^{-01}$)	6.87 (1.36 $\times 10^{-11}$)	6.15 (1.23 $\times 10^{-09}$)	
Pitch Direction Changes	3.53 (4.25 $\times 10^{-04}$)	-0.02 (9.86 $\times 10^{-01}$)	5.98 (3.55 $\times 10^{-09}$)	5.92 (4.79 $\times 10^{-09}$)	
Intervallic Leaps	3.89 (1.07 $\times 10^{-04}$)	1.95 (5.11 $\times 10^{-02}$)	4.7 (3.07 $\times 10^{-06}$)	2.75 (6.12 $\times 10^{-03}$)	
Root Notes	4.45 (9.26 $\times 10^{-06}$)	3.51 (4.73 $\times 10^{-04}$)	3.95 (8.67 $\times 10^{-05}$)	0.37 (7.08 $\times 10^{-01}$)	

Table 3.4: Table showing t and p values of t-tests with $\alpha = 0.05$, between the PAF and the ANOMIC (musicians and non-musicians) datasets on the features of the first column. Orange cells denote tests with $p > 0.05$, which means that the difference between the two tested distributions is not statistically significant.

eral features, based on how intuitive they are and their correlations, and narrowed the list down to 7 features: pattern duration, occurrences, last note duration, note range, pitch direction changes, intervallic leaps and root notes. These features are also listed in Table 3.3, along with their descriptions. Plotted distributions of these features can be seen in Figure 3.5. A list of all analysed features, all t-test and KS-test results as well as all boxplot visualisations are available online⁵.

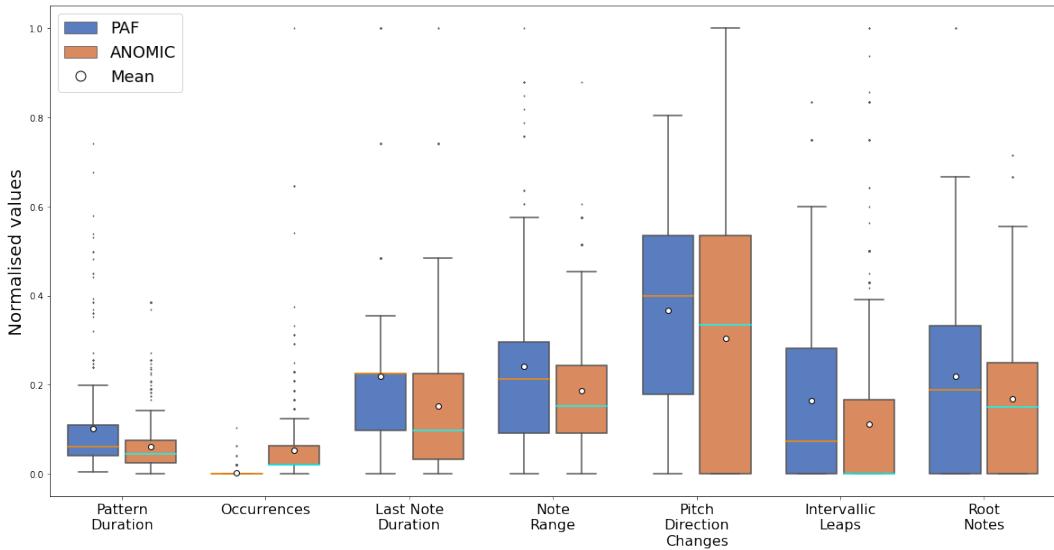


Figure 3.5: Boxplots showcasing the distributions[fn:results] of the analysed pattern features of the PAF and the ANOMIC datasets. For each feature, the two distributions were normalised.

Duration and Occurrence features

The first of the features we analysed was the **Pattern Duration** feature, which measures the length of a pattern in quarter notes (crotchets). We observed that the ANOMIC distribution had much smaller overall and interquartile ranges than the PAF distribution. The mean and median values of the distribution were also much smaller.

Next was the **Occurrences** feature, which refers to the number of times that a pattern occurs in a music excerpt, as defined by (Collins, 2011). From the boxplots in Figure 3.5, it is evident that the PAF distribution differs drastically from the ANOMIC distribution, which is expected due to the difference in instructions given about whether to annotate all occurrences or not. This result confirms that the annotators followed the instructions closely in this respect.

Differences between the two datasets were also seen for the **Last Note Duration** feature, which describes the duration of the last note of a pattern. Here, the

ANOMIC distribution had larger interquartile and overall ranges than the PAF distribution, while its mean and median values were lower.

The t-tests in Table 3.4 and the KS-tests (available online⁵) confirmed that the above-mentioned differences between distribution pairs were statistically significant. Furthermore, they revealed that the differences between the PAF dataset and the two subgroups of the ANOMIC dataset, based on musical backgrounds, were also statistically significant. Thus, we may conclude that the differences between feature distributions of pattern datasets were caused by varying musical backgrounds but also possibly by the interfaces or the instructions of the annotation tools.

Pitch-based features

The two annotation datasets also differed on the **Note Range** feature, which describes the number of semitones between the lowest and highest note of a pattern. Figure 3.5 shows that the PAF distribution has a much larger overall and interquartile range than the ANOMIC distribution. Its median and mean values are also higher.

Similar differences were also seen for the **Pitch Direction Changes** feature, which is defined as the number of melodic arcs in a pattern (inspired by various melodic arc features in (van Kranenburg *et al.*, 2013)). Our analysis showed that the ANOMIC distribution had a much larger overall and interquartile range. Furthermore, the distribution lacks the bottom whisker, thus being more positively skewed. The ANOMIC distribution also had significantly lower mean and median values.

We observed comparable differences among the **Intervallic Leaps** feature distributions. The feature describes the fraction of all melodic intervals of a pattern whose note range is larger than two semitones, as defined by (Collins, 2011). Considering the analysis results, we see that the PAF distribution has much larger overall and interquartile ranges. Its median and mean values are also significantly higher.

Finally, differences were seen for the **Root Notes** feature, which describes the fraction of notes in a pattern that are root notes or octaves of roots of the music piece. From the distributions present in Figure 3.5, we can discern that the distributions mainly differ in the overall and interquartile ranges, with the PAF distribution having much larger ranges than the ANOMIC distribution. We also note that the mean value, as well as the standard deviation of the PAF distribution, are slightly higher than those of ANOMIC.

By analysing the t-test results from Table 3.4 and the KS-test results (available online⁵), we confirm that the distribution differences of the first three pitch-based features between the two datasets are statistically significant. We notice that these three features pertain to the relationships between notes in the patterns

and are therefore melodically relevant. We can also discern that the differences are not statistically significant between the PAF group and the musician group of ANOMIC. However, the difference between musicians, both PAF and ANOMIC, and non-musicians of ANOMIC, is statistically significant.

3.5 Conclusion

Based on our comparison results between PAF and ANOMIC in the last section, we draw a few conclusions in this section. Comparing the annotations collected with the PAF tool enabled us to study three groups of annotators from different musical study programmes. We observed higher agreement between annotators of the same group when compared to annotators of different groups, indicating a potential influence of study programmes on the understanding and perception of patterns in music. Comparing the annotations of the ANOMIC annotation set, which included two groups of annotators (musicians and non-musicians), gave a similar result.

Several differences observed in the feature-based comparison of the annotation datasets point towards discrepancies in annotations caused by design differences in the tools and experiments. The PAF interface displays sheet music allowing a compact representation, with large sections of the music piece being presented to the user within a single view. By contrast, ANOMIC's piano roll representation generally displays fewer elements at once to the user in order to preserve element clarity. The difference in music visualisation might have caused users to perceive and annotate patterns with durations relative to the view window size. The notes in the sheet music representation of PAF remained roughly the same size, while the piano roll elements of ANOMIC varied in their size based on the durations. Since the last note durations of the PAF dataset are on average significantly longer, differently sized elements of the ANOMIC tool could have discouraged users from picking longer notes as pattern endings. The large disparity in the occurrences feature distributions was most likely caused by the lack of automatic occurrence matching functionality in the PAF tool, as well as differences in the instructions given to the users, despite the augmentation step as described in Section 3.2.6.

Moreover, we observed differences in note range, intervallic leaps, and pitch direction changes between ANOMIC non-musicians versus ANOMIC musicians and PAF annotators. The t-test analysis showed that the differences were not statistically significant between the PAF group and the musician group of ANOMIC. However, the difference between musicians, both PAF and ANOMIC, and non-musicians of ANOMIC, was statistically significant. We, therefore, conclude the varying musi-

cal background of the ANOMIC annotators as the underlying cause, and exclude the potential influence of the annotation tools. We also speculate that annotators with different musical backgrounds tend to annotate patterns with different melodic properties. Finally, for the root notes feature, we are unsure what might have caused the difference in distributions, though we believe it is not caused by the musical background based on the lack of difference between feature distributions of the ANOMIC musicians and non-musicians.

Our findings point to a major influence of the annotation tools, instructions, and the musical background of participants on the annotated patterns. As a next step, the influence of the tools should be studied in more detail using stricter controlled comparisons, including a clarification on how users should include their listening experience into the annotation process, and a controlled use of the automatic pattern matching functionality. Moreover, the analysis of the pattern datasets can be enriched by further investigations as to where annotators tend to agree, for instance, by exploring dynamic thresholds for calculating inter-annotator agreement depending on the size of the patterns. Determining in more detail different levels of granularity as to when two pattern annotations can be considered as agreeing, even if the exact beginning and ending points are not identical, can further help to identify different layers of commonality between annotators.

We believe that the widespread use of digital tools in gathering pattern annotations is inevitable in the near future. Our findings point to several directions for improvements in large-scale data collection and analysis of musical patterns. The observed differences in annotations gathered with different tools call for further experiments and analyses for deriving technical design choices that fit the purpose of pattern annotations in an optimal way for a given context and annotator group. Establishing reference data for evaluating automatic pattern discovery algorithms from such rich annotation datasets can follow different directions. For instance, identifying subgroups of annotators that highly agree with each other can assist in establishing single-reference data based on a larger group of annotators. Establishing evaluation methods that take into account multiple reference annotations expressing different subjective interpretations of the same musical piece, can pave the way for a more adequate consideration of ambiguity and subjectivity in the evaluation of pattern discovery algorithms.

3.6 Summary

This chapter described and compared two annotation tools and three annotation experiments using the same musical pieces. The annotations were gathered using different annotation strategies but with a similar description of what a musical pattern is. We digitised the data from one of the experiments and used inter-annotator agreement analysis as well as feature-based analysis on the other two experiments. The two analyses revealed differences in the annotated patterns between user groups and between annotation tools and their instructions.

In an attempt to objectify a relatively subjective task, such as annotating patterns, the influence of a variety of factors are often difficult to untangle. In the described experiment, we addressed factors such as the annotation interfaces, instructions, and user backgrounds. However, we hypothesise that there are other latent factors, which have yet to be discovered. In using them for evaluating pattern discovery algorithms, we should be careful about designing and testing the metrics coming out of annotation data. Devising a new metric and testing its efficacy is beyond the scope of this work; however, it opens several directions for future work.

3.7 Discussion

We have discussed several factors to consider when collecting human-annotated musical patterns for algorithms. For the public's and experts' consumption, the data and algorithms both need to be carefully deliberated.

In this section, we discuss more points of our limitations and future work. We will also touch on topics that are tangentially related to annotation experiments. We have quite a few points in this chapter in comparison to other chapters, as experiments involving humans can be more convoluted.

The control of variables: instruction, visualisation, and helper functions

In this first explorative study of comparing annotation tools, we did not provide a single set of instructions for musical experts (PAF) as well as non-musicians (ANOMIC). Giving instructions to the annotators is one of the most difficult aspect of pattern discovery experiments to design, because we have a flexible concept of pattern and are trying to find something that we do not know ahead of time. We have not used the working definition given in Chapter 2 because we followed the instruction from previous work in this series of experiments. Additionally, we cannot verify if the annotators strictly adhered to the instructions. In the future,

including more specific instructions, incorporating our definition that emphasises the importance of why a pattern is annotated, and adding user behaviour logging may allow for more controlled conditions when comparing different tools.

For instance, an instruction to first listen to music without consulting the visualisation before starting with the annotation process, might decrease an otherwise perhaps strong tendency of users to annotate patterns they can visually identify. Nevertheless, each music visualisation will influence the annotation process to a certain extent. Musical experts may be most familiar with sheet music, but piano roll visualisations may be more accessible for annotators with less musical expertise. As our results indicate that the size of the musical excerpt that can be displayed in a single view to the user seems to influence the length of patterns annotated, this should be specifically considered when longer patterns are expected to be important for a specific corpus.

This instruction issue is related to the helper functions as well. The influence of the automatic occurrence matching functionality needs to be investigated in more depth in the future. If the goal of the annotation is to find all occurrences of a given pattern, as was the case in the ANOMIC experiment, it can alleviate the finding of the pattern occurrences for users, but might have the side effect of pointing users to occurrences they would not have deemed important otherwise. If only the most representative pattern should be annotated, as in the PAF experiment, such a tool can assist in highlighting all other occurrences from which the user can then choose the most representative one for the annotation. Either way, this calls for a systematic investigation of using tools with and without such functionality.

The helper function embodies this cycle of improving pattern finding algorithms and collecting annotations: the algorithm can help with the data collection process, which helps to improve the algorithm itself. In fact, it is becoming increasingly common to use algorithms as pseudo-annotators for data collection and self-supervised learning in [ML](#) (Carmon *et al.*, 2019). Another possibility to explore is to create a tool that enables annotators to write their own helper functions based on their domain knowledge using a simple [DSL](#), which is also an active strand of research (Ratner, Bach, *et al.*, 2017). [DSLs](#) will be discussed more in detail from Chapter 5 onwards.

Measures of musical background

Fairly assessing and comparing different music backgrounds can be a challenge. For most of us, musical ability exists primarily on a spectrum, and drawing a clear line between musicians and non-musicians is difficult.

In this chapter, we used either the faculty programme of the participants or a basic questionnaire to approximate level of musical expertise for our experiments. We did not collect other detailed music background of the participants, such as what music they usually listen to, whether they have perfect pitch, whether they studied the pieces in experiments before. In future experiments, more of this information and sophisticated musical expertise indexes could be used (Müllensiefen *et al.*, 2014). Asking about the participants' familiarity with the music material can be potentially useful as well in our analysis.

Static threshold

The threshold in our agreement metric can be controversial. When the threshold is large, disagreeing parts of annotations could be longer than the overlapping parts, while the two different pattern annotations could still be evaluated as agreeing.

This is possible indeed, for example, if the annotations in questions are beat 17-21 and beat 20-23. In general, using a_1, a_2, b_1, b_2 to represent the boundary of the patterns, these cases occur when $5 > |a_1 - a_2| + |b_1 - b_2| > |a_2 - b_1|$, $a_1 < b_1 < a_2$. Solutions we can find when $\text{threshold} = 5$ include the annotated patterns with an overlap of 0-3 beats. These are indeed difficult edge cases with large threshold values. However, there always exists a certain degree of proximity in which the patterns appear when the measures are of high values. In addition, it does not seem to happen often in our dataset, as can be seen from the results using threshold value 1. As we mentioned in Section 3.3, this static thresholding approach should be improved in the future.

Modelling experts

Our participants are mostly bachelor's and master's students, and none of them are academic experts, such as professors of musicology or music theory. For developing algorithms, two specific types of humans-in-the-loop—crowd-in-the-loop or experts-in-the-loop—have different costs. The higher the expertise, the harder it is to perform annotation experiments with them due to limited availability. Experts' annotations, too, can disagree and be associated with another set of challenges. Nevertheless, a single expert's idea is important, and one may employ a different approach to computationally model an expert.

It is likely that the annotations from a single expert would be smaller in numbers in comparison to a group of people, and therefore require approaches from the tradition of the humanities. It could be similar to what we have seen in Chapter 1

that, despite the eloquent characterisations of pattern-related concepts in previous research, they are too imprecise and unsystematic for computational approaches. Big data and machine learning approaches, except zero-, one-, n-/few-shot learning, would likely to fail when the dataset is small, too.

If there exists a well-defined, mathematically or computationally inclined musical theory of patterns where a set of axioms and rules are explicitly given and fixed, then a different approach than the data-driven ones could be computationally implemented. In this type of top-down modelling, the evaluation would consist of testing and verification of the system in order to guarantee the correctness of the implementation, instead of comparing with annotations. However, it may deliver extra insight to try to align these top-down theories with listening experiences that involve the bottom-up process of pattern annotation by humans.

Negotiation and consultation

In our experiments, the participants could not talk to each other and modify each other's annotations according to the discussions. It is possible that, after being in consultation with each other, the annotators may change their mind, or even agree to disagree. In fact, this is related to the topic of making adaptive or instinctual choices. The potential for disagreement stems from the indecision in selecting between the option that "feels right" and the one that is more deliberate. The problem is complicated even more if we allow for these different levels and types of disagreement as well.

Future experiments can consider this approach and incorporate a chat or vote functionality in the annotation tools. We believe, however, that there are merits in looking at the first reactions from annotators without modifications.

Intra-annotator (dis)agreement

In this chapter, we did not touch on the topic of intra-annotator (dis)agreement. However, this type of disagreement probably happens often in the case of musical patterns: the same annotator might disagree with their past selves. In fact, we have reached out to the annotators for the HEMAN experiment after five years since the experiment took place, but did not receive a reply from them. It would be interesting to consider a longitudinal study in the future.

In comparison with the study of disagreement in chord estimation

(Koops *et al.*, 2019) investigated the inter-annotator agreements in chord estimations and was inspirational for this chapter. Here, we make a comparison between the pattern discovery task and chord estimation task in terms of the presence of multiple annotations.

In chord estimation, there is a chord or a group of chords to be assigned to specific points or periods in music. In pattern discovery, the assignments are not local but global. For every point of the music, the annotator can point out whether there is or there is no pattern. When pointing out there is a pattern, this point automatically relates to another or many other points corresponding to the pattern's occurrences in the music. Therefore, although both chord estimation and pattern discovery face inter-annotator agreement issues, the nature of the problems is different.

Disagreement and uncertainty

It has been said in economics that when there is disagreement, there is uncertainty (Bomberger, 1996). Let us take a look back into the deeper implications of agreement and disagreement in terms of certainty and uncertainty.

Why do we need multiple annotators in the first place? Because there is no reason to expect that they will be in perfect agreement, and evidence shows that this is indeed not the case. We have already seen that there can be a variety of reasons as to why annotated patterns differ. Disagreement reflects inter-personal and, in some cases, intra-personal differences that give rise to uncertainties. Uncertainty makes it challenging to define patterns extensionally (see Section 2.2 where we discussed intension and extension definitions.)

There are unambiguous situations where well-defined notions guarantee agreement, such as the task of simple counting or calculations, which often do not require more than a very small number of people to perform the task. The regions in-between the certain/uncertain extremes consist of a spectrum of possibilities, where it is hard to tell whether a definitive answer always exists or uncertainties dominate.

In either case, facing uncertainty, we can take the intersection of the annotations, or the aggregate, or sieve through them according to different levels of importance. In the area of musical pattern discovery, since there is a lack of annotated data, the default has been to take the aggregate of data. As suggested in Section 3.5, other strategies should be investigated, too.

Data-centric and model-centric views

In the machine learning community, there has been a discussion about data-centric and model-centric views. Although that the data-model interplay does not naturally bias one or the other, it is a fact that a majority of research puts focus on the models and algorithms, instead of the data they operate on. However, as suggested in (Northcutt, Jiang, *et al.*, 2021), a data-centric model may result in a more efficient and meaningful solution. An example of this would be an iterative data gathering process where the data get improved and the models are kept fixed.

While we believe that both data and models are essential, we agree that training data has become a key differentiator for the performance of algorithms (Ratner, Bach, *et al.*, 2017), and that this data should not be kept fixed. In fact, as gathering data is a messy process, major errors are found in widely used datasets (Northcutt, Athalye, *et al.*, 2021). Data should not become the new "Bible" for truth, but it serves as a reference that could be investigated thoroughly and progressively.

Chapter 4 Comparisons and Evaluation of Musical Pattern Discovery Algorithms

The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly. A programmer is ideally an essayist who works with traditional aesthetic and literary forms as well as mathematical concepts, to communicate the way that an algorithm works and to convince a reader that the results will be correct.

– Donald E. Knuth

We explored human-annotated musical patterns in the previous chapter, and in this chapter, we compare between these human annotations and algorithms as well as the algorithms themselves. We make these comparisons to reveal that musical pattern discovery algorithms face the following challenges:

- Patterns discovered by different algorithms for the same piece differ greatly,
- The output patterns are often difficult to relate back to meaningful musical concepts,
- The output patterns do not agree well with human annotations,
- Different application contexts for musical patterns might require different types of patterns.

To address these challenges, we introduce two visualisation approaches and two computational methods for examining algorithmically discovered musical patterns: Pattern Polling ([PP](#)), to combine the patterns, and Comparative Classification ([CC](#)), to differentiate the patterns. In addition, we propose to plant predetermined patterns into random data to generate controllable synthetic data, thereby leaving us better able to inspect, compare, validate, and select the algorithms. We provide a concrete example of using synthetic data for understanding the algorithms and ex-

pand our discussion to the potential and limitations of such an approach. Finally, we will finish the main content of the chapter with a discussion over the ground truth problem, summarise the chapter, and discuss topics related to pattern discovery algorithms in general.

4.1 Overview

Music and patterns in music provide unique opportunities and challenges to understanding and using algorithms. For the task of musical pattern discovery, we have seen in Chapter 1 and 2 that patterns are ubiquitous. Algorithmic musical pattern discovery research aspires to uncover and extract such elements automatically.

Research into musical patterns and pattern discovery algorithms is of current relevance (Collins, 2011; Collins *et al.*, 2013; Conklin, 2002; Conklin & Anagnostopoulou, 2011; Forth, 2012; Hsu *et al.*, 1998; Janssen, 2018; Janssen *et al.*, 2014; Lartillot, 2004; Meredith, 2013, 2015; Meredith *et al.*, 2002; Nieto & Farbood, 2014; Pesek *et al.*, 2017; Ren, 2016; Ren *et al.*, 2017; Ren, Volk, *et al.*, 2018; Rolland, 1999; Velarde *et al.*, 2016; Velarde *et al.*, 2013). With recent and rapid development in research areas such as pattern recognition and machine learning, many more **ML** algorithms that are potentially suitable for automatically extracting musical patterns have become available. Together with the curiosity-driven pursuit of unveiling hidden structures in music data, algorithmic musical pattern discovery inspires many to explore and understand music from its modular components and their combinations. Another motivation driving the development of musical pattern discovery is the potential for applications in areas such as: genre classification, error correction, automatic transcription, and segmentation in **MIR** (Cambouropoulos, 2006; Conklin, 2010; Conklin & Anagnostopoulou, 2006; Dixon *et al.*, 2004; Lin *et al.*, 2004); as well as automatic composition and learning systems in **HCI** (Collins, 2011; Fowler, 1966; Kaplan, 2015). For example, given that composers employ patterns to introduce structure into their music (Hsu *et al.*, 1998), the discovered patterns can be used to provide auto-completion and inspirational suggestions; we can also highlight discovered patterns in music as a guide for attentive listening, memory anchoring, and efficient practising (Jones, 1987; Kubik, 1979); for musicologists and music theorists, facing the complexity of large corpora, the algorithmically discovered pattern candidates can provide support and evidence for categorisation and theorisation efforts (Agawu, 2014; Gjerdingen, 2007; Huron, 2006; Lerdahl & Jackendoff, 1985; Zbikowski, 2002).

Challenges from the algorithms

In spite of the existence of many generic pattern discovery algorithms (Bertens *et al.*, 2016; Brand, 1999; Cooley *et al.*, 1997; Papadimitriou *et al.*, 2005; Parida, 2007; Vreeken *et al.*, 2011; Wang *et al.*, 1994) and musical pattern discovery algorithms (Collins *et al.*, 2013; Conklin, 2002; Conklin & Anagnostopoulou, 2011; Forth, 2012; Hsu *et al.*, 1998; Larillot, 2014; Meredith, 2013; Nieto & Farbood, 2014; Pesek *et al.*, 2017; Ren, 2016; Rolland, 1999; Velarde *et al.*, 2016; Velarde *et al.*, 2013) (see (Janssen *et al.*, 2014) for a detailed overview), there nevertheless remain some persistent challenges. In addition to the difficulties we addressed previously, such as the definition problem and the ambiguity and subjectivity, there are two more inter-related and algorithm-specific ones. First, there can be a large number of output patterns, which are costly to examine manually. Second, implementation logic can be hard to comprehend, which could be caused by any of the numerous procedures that comprise the algorithm, or by a binary-only release for which we only have access to the output. Automating musical pattern discovery is not yet a solved problem.

Comparisons between the algorithms

Comparing between the algorithms are tricky. Using methods inspired by geometric shapes, machine learning, bioinformatics and more, pattern discovery algorithms were previously tested on unassociated datasets with disparate metrics (Janssen *et al.*, 2014). One attempt to standardise the evaluation of algorithms is the [MIREX](#) task we described in Chapter 1. In the task, a *pattern* is defined as a set of time-pitch pairs that occurs at least twice in a piece of music and the human-annotated [The Johannes Kepler University Patterns Development Database \(JKU-PDD\)](#) dataset was introduced (“2014:Discovery of Repeated Themes & Sections - MIREX Wiki”, 2014). Metrics were devised to calibrate the algorithmic output, but algorithms rarely performed consistently on all the testing pieces. Patterns extracted by different algorithms can also vary to a great extent given the same input, which creates controversy when it comes to using human annotations as ground truth and when designing an all-encompassing evaluation strategy (Janssen *et al.*, 2014; Meredith, 2015; Ren *et al.*, 2020; Ren *et al.*, 2017).

Another pattern annotation dataset which has been used for evaluating the algorithms is the [MTC-ANN](#) dataset (Boot *et al.*, 2016; van Kranenburg *et al.*, 2016). In contrast to a direct comparison such as in MIREX, algorithms’ output have been compared in a classification and compression task in (Boot *et al.*, 2016) together with human annotations. Results show that the human-annotated patterns perform better than the algorithms. Both metrics in MIREX and (Boot *et al.*, 2016) are difficult to

translate back to a straightforward explanation of the algorithms, or points that can be used to improve the algorithms.

Building on this prior research, our intention of comparing the algorithms with human annotations is not to claim any superiority between annotations and computed patterns, and between the algorithms, but to help one another to find meaningful patterns. Figure 4.1 shows a diagram of some possible interactions between the human annotators, algorithms, and the tasks where patterns are to be employed. The ultimate purpose for comparing algorithms in reference to human data is to evaluate which algorithms are more suitable and more similar to humans in certain situations or tasks. In this dissertation, we do not produce an answer for this ultimate question, but propose methods for examining this suitability.

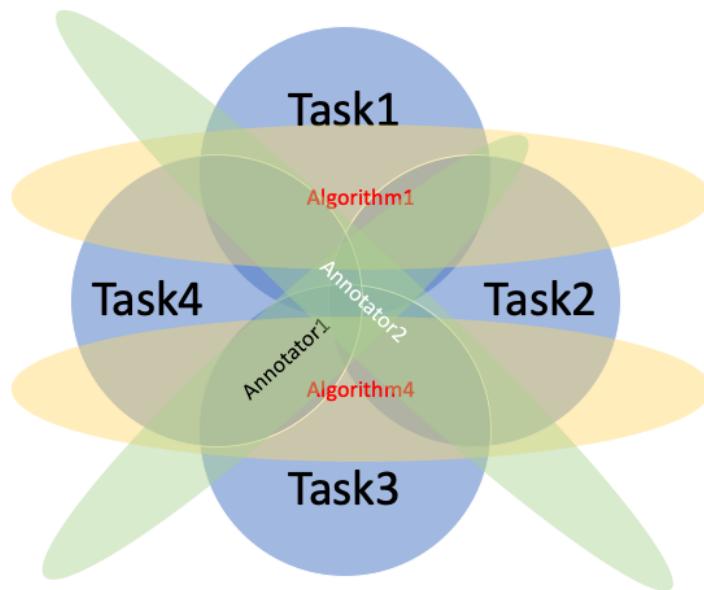


Figure 4.1: An example of the interactions between tasks, annotators, and algorithms. The overlap between the tasks and agents (an annotator or an algorithm) signify that they have high compatibility.

Summarising the use of the algorithms in our subsequent experiments

Table 4.1 summarises how we use some of the algorithms we introduced in Chapter 2 in our subsequent experiments, and whether they have been submitted to the MIREX task. The visualisation, PP, CC, and Synth experiments will be seen in Section 4.2.2, 4.2.3, 4.2.4, and 4.2.5 respectively.

Some algorithms are not present for all experiments because of format compatibility. Not including all algorithms does not undermine the validity of our results because we use the algorithms individually and provide our analysis mainly on a

case-by-case basis, that is, we will look at algorithms in a modular fashion. We expect that our analysis and methods could still be applied in the presence or absence of any number of other algorithms.

Algorithm	MIREX	Vis.	PP	CC	Synth
VM1	x	x	x	x	x
VM2	x	x	x	x	x
SIAF1 (SIATECCompress - F1)	x	x	x	x	x
SIACR (SIATECCompress - Recall)	x	x	x	x	x
SIACP (SIATECCompress - Precision)	x	x	x	x	x
SC	x	x	x	x	
OL1 (PatMinr)	x	x	x		
OL2 (PatMinr)	x	x	x		
ME (Motive Extractor)	x	x	x		
SIACFP	x	x	x		x
MGDP					x
COSIATEC					x
Forth					x

Table 4.1: Table summarising the musical pattern discovery algorithms we examine. The "MIREX" column indicates whether they have been submitted to the MIREX task. "Vis." denotes visualisation. The rest of the three columns indicate whether we consider them in our three other experiments. A cross mark indicates that the algorithm is present.

This chapter

This chapter is based on our recent work on leveraging different visualisation methods and classification algorithms (Ren *et al.*, 2017; Ren, Volk, *et al.*, 2018) to show differences between human annotations and the output of [SotA](#) musical pattern discovery algorithms. Following a similar direction, this chapter aims to analyse disagreement between algorithmically extracted patterns and human-annotated patterns and between pattern discovery algorithms themselves. Our intention is not to create metrics to rank the algorithms, but to introduce methods to examine their output, which could potentially inform us more about the algorithms as well as the music data being used. In the following text, we provide an introduction, comparison, and evaluation of musical pattern discovery algorithms, propose new visualisations, two new methods for combining and distinguishing discovered patterns, and finally discuss the use of synthetic data for evaluating the algorithms.

4.2 Our approaches

In this section, we represent a few ways to computationally inspect, compare, verify and evaluate several musical pattern discovery algorithms. We start with an overview with the metrics used in the MIREX task to see how we can improve on the [SotA](#). We will see that we can improve in five directions: visualisation, combining the algorithmic output, classifying the output, using synthetic data as an application of the patterns. In finishing, we will discuss the ground truth problem that is at the heart of any evaluation effort.

4.2.1 MIREX metrics

We mentioned in Section 3.3.1 that the MIREX metrics take into account actual musical events. Here is a list of all the metrics used in the task:

- Establishment precision, establishment recall, and establishment F1 score
- Occurrence precision, occurrence recall, and occurrence F1 score
- Three-layer precision, three-layer recall, and three-layer F1 score
- Coverage and compression ratio
- Runtime, fifth return time, first five target proportion and first five precision
- Standard precision, recall, and F1 score
- Friedman tests

The establishment and occurrence metrics focus on two different points for pattern discovery. Establishment metrics measure whether an algorithm is capable of establishing that a pattern is repeated at least once during a piece, and are less interested in whether the algorithm can retrieve all in/exact occurrences. Occurrence metrics focus on an algorithm's ability to retrieve all occurrences. These two aspects are not completely orthogonal to each other, because occurrences help in establishing the patterns.

Three-layer metrics cross-compare all discovered occurrences with ground-truth occurrences. Friedman tests are chosen to investigate whether any algorithms rank consistently higher or lower than the others regarding metrics for individual pieces. Coverage and compression ratio concern with the relationship between the number of notes in the pattern discovery output compared to the set of notes in a musical work. Coverage is the percentage of notes in the musical piece that is covered by discovered patterns. Compression ratio computes how far a musical work can be more efficiently expressed in discovered patterns and their occurrences. Formal defini-

tions are available in (“2017:Discovery of Repeated Themes & Sections - MIREX Wiki”, 2017).

Most evaluation metrics rank the algorithms according to a score as the proxy for how meaningful the discovered patterns are, and this is no exception for MIREX. However, these metrics are not entirely without controversy: the performance of algorithms varies on different metrics and different musical pieces, for example, making it difficult to analyse the overall and specific strengths and limitations of algorithms. Problems such as these motivate us to examine the output patterns from the algorithms in more detail to advance this area of research.

4.2.2 Visual comparison

In this section, we compare discovered patterns visually in two different ways. We first give a short introduction to visualisations in research in general. Following that, we visualise algorithmically extracted patterns and human-annotated patterns using pattern positions and then pattern features. The challenge here is to visualise a large number of discovered patterns in relation to each other. In answer to this, we abstract away different aspects in different visualisations. In later chapters, we also consider interactive visualisations where the inspector can zoom in and out of different aspects.

Visualisation in general

“Provare per credere”—seeing is believing—is a concept perhaps not foreign to many. “Visualisations help offload cognition to perception,” says Jessica Hullman. It has been demonstrated that visualisation can assist humans in debugging complex reasoning steps (Shams *et al.*, 2018). (Tukey & Wilk, 1966) also stated that the effective laying open of the data to display the unanticipated is a major portion of data analysis. Visualisation is an effective tool for data storytelling, a way of making complex information relatable and different perspectives shareable. Visualisation from data enables us to straightforwardly compare our expectations and the data, which can be used as powerful as a type of pseudo-statistic model fitting and checking. By targeting the human visual cortex, we can encourage discovery with a variety of visual cues.

What is the best way to organise visualisation for presenting musical patterns? There are many possibilities, including static figures, score highlighting, interactive applications, animated films, to list a few. Considering the simplicity and the static

nature of most academic publications, we start with static visualisations in this chapter.

Visualising pattern positions

As described in Chapter 2, the beginning and endings of patterns are perhaps the most important to show, because they determine the musical content when the piece of music is fixed. Therefore, using these two values, we can concisely encode and represent a musical pattern occurrence within a musical piece. In Figure 4.2, we visualise and emphasise the (non)existence of patterns by plotting a bar in between the beginning and ending points of the pattern occurrences. In this way, we can concisely show a large number of patterns and their relations to each other in position.

The music data used in this figure is the monophonic version of Chopin’s Mazurka Op.24 No.4. The algorithms used are summarised in Table 4.1, with a slight change in the naming convention: SIAF1, SIAP, SIAR, in the figure are SIACF1, SIACP, and SIACR, correspondingly.

We can make several observations from Figure 4.2:

- Different algorithms find very different patterns—some tend to find shorter patterns, some longer; some find many patterns while others are more discerning.
- We have three algorithm families (SIA, VM, and OL), each of which consists of more than one algorithm. Algorithms from the same algorithm family tend to find similar patterns. Similarities observed include the number of patterns discovered, coverage of the song, and occurrence overlaps.
- The ground truth (which we take to be human annotations) is sparse in comparison to the patterns discovered by the algorithms.
- Taking an overview of the entire visualisation, we can see some correspondence and similarities between the algorithms and the ground truth patterns.

This type of visualisation might remind the reader of the orchestral graphs from (Dolan, 2013), as shown in Figure 4.3. Although the appearance is similar—with horizontal bars at different positions on the y-axis and time on the x-axis, what the bars represent is quite different. In our case, each bar represents the existence of a pattern, and there can be a large number of bars; in the case of orchestral graphs, the bars represent the existence of instruments sounding in the music, and the number of bars does not exceed the number of how many types of musical instruments are present. Our visualisation is developed independently from this work.

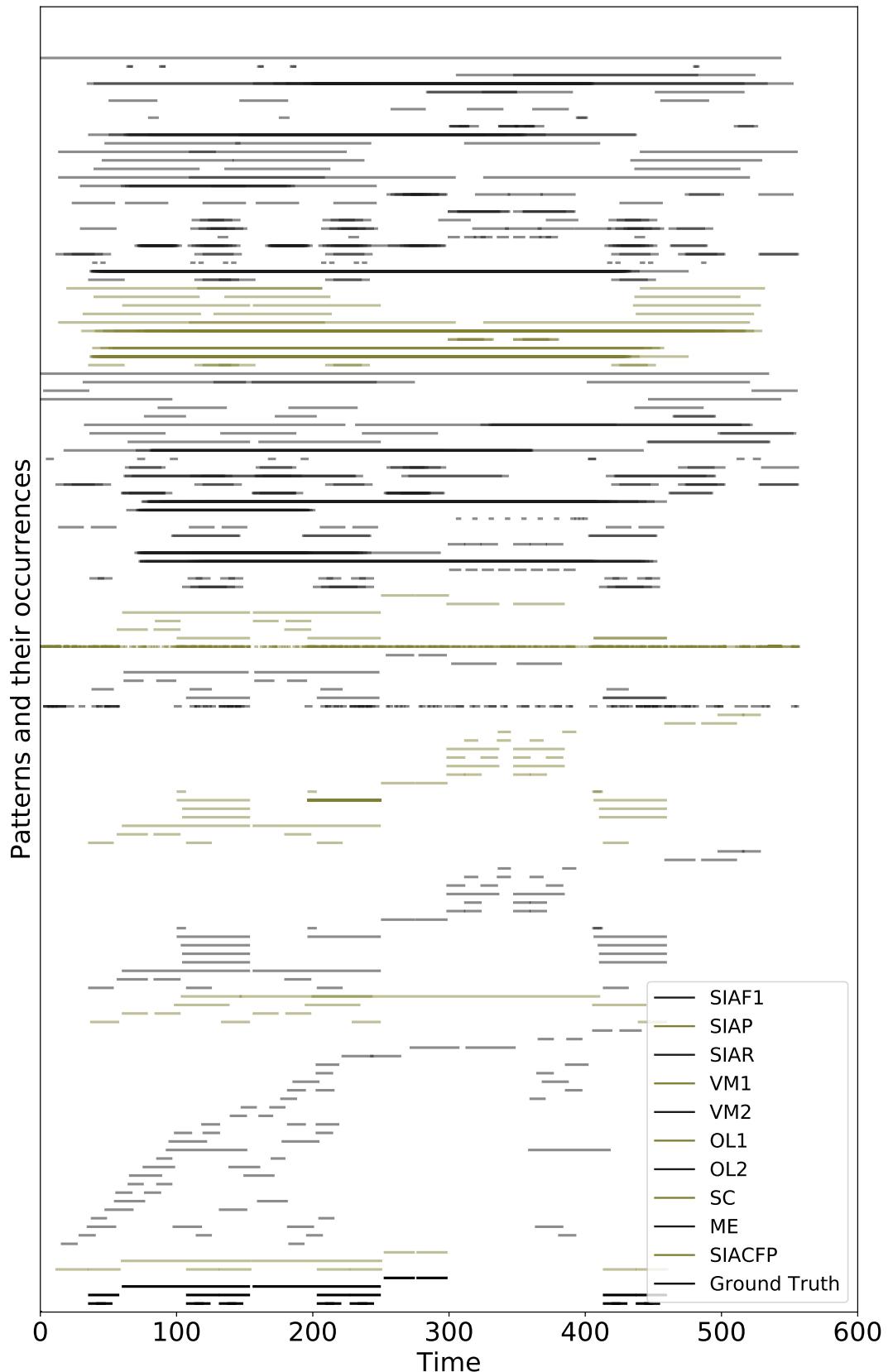


Figure 4.2: Patterns extracted by all algorithms submitted to the [MIREX](#) task 2014-2016 plus [SIACFP](#) on the monophonic Chopin's Mazurka Op.24 No.4. A horizontal bar shows the presence of a pattern. The x-axis represents the time offset in crotchet units. We can see the algorithms find different numbers of patterns and patterns of different lengths.

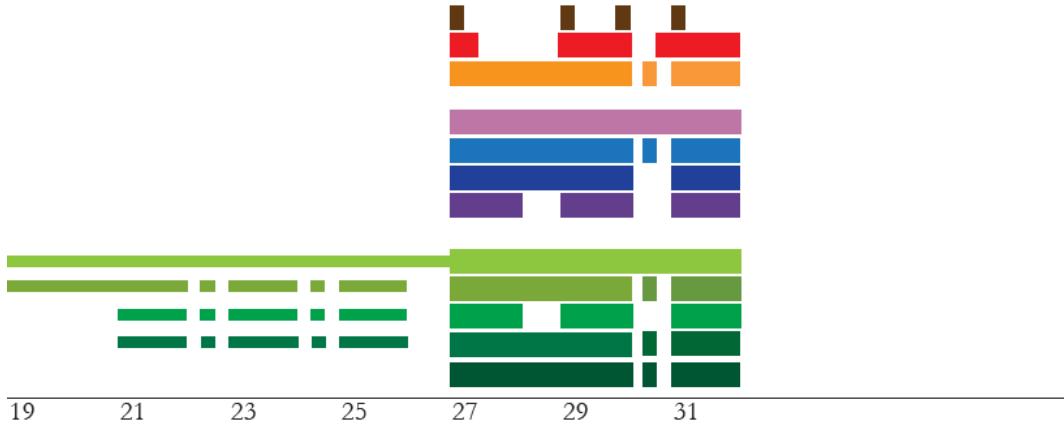


Figure 4.3: An example of orchestral graphs. Taken from Dolan, 2013.

Using the same visualisation method, we can also visualise patterns across different pieces. In Figure 4.4, we showed the positions of human-annotated patterns in the [MTC-ANN](#) dataset¹ with which we plot 24 out of 26 tune families and numerous human annotations as can be seen in the figure. Each vertical height contains the patterns from a different piece. By putting different pieces together, we will see the difference between the lengths of different pieces: some pieces are longer and some are shorter. Although this cross-piece comparison may not work on vastly different pieces of music, it is fitting for the [MTC-ANN](#) dataset. We can observe that, in the same tune family, it is very likely to have patterns in similar locations.

Taking Figure 4.4 further, we attempt to show the basic pitch and rhythm information of the patterns in Figure 4.5. Each dot represents a musical event, in which the rhythm is shown by the spacings of the dots and the pitch differences are reflected in very slight vertical offsets. Although we can glean some useful information visually, such as the fact that some inserted notes can be seen in several pattern occurrences, there is a limit to this type of visualisation: the pitch differences are difficult to discern and confusing given the restricted spacing of each piece. Without an interactive zoom-in functionality, this is difficult to improve. In future work, interactive visualisation methods could be desirable where both the macro- and micro- scale of the musical content can be shown and explored. We also show an interactive visualisation in a different context in Section 7.3.6.

Visualising using MDS

[PCA](#) is a well-known feature-extraction and [MDS](#) method for visualisation. It outputs combinations of features that form orthogonal principal components. These principal components are in the same directions as the directions of the

¹More details about the dataset can be found in Appendix A

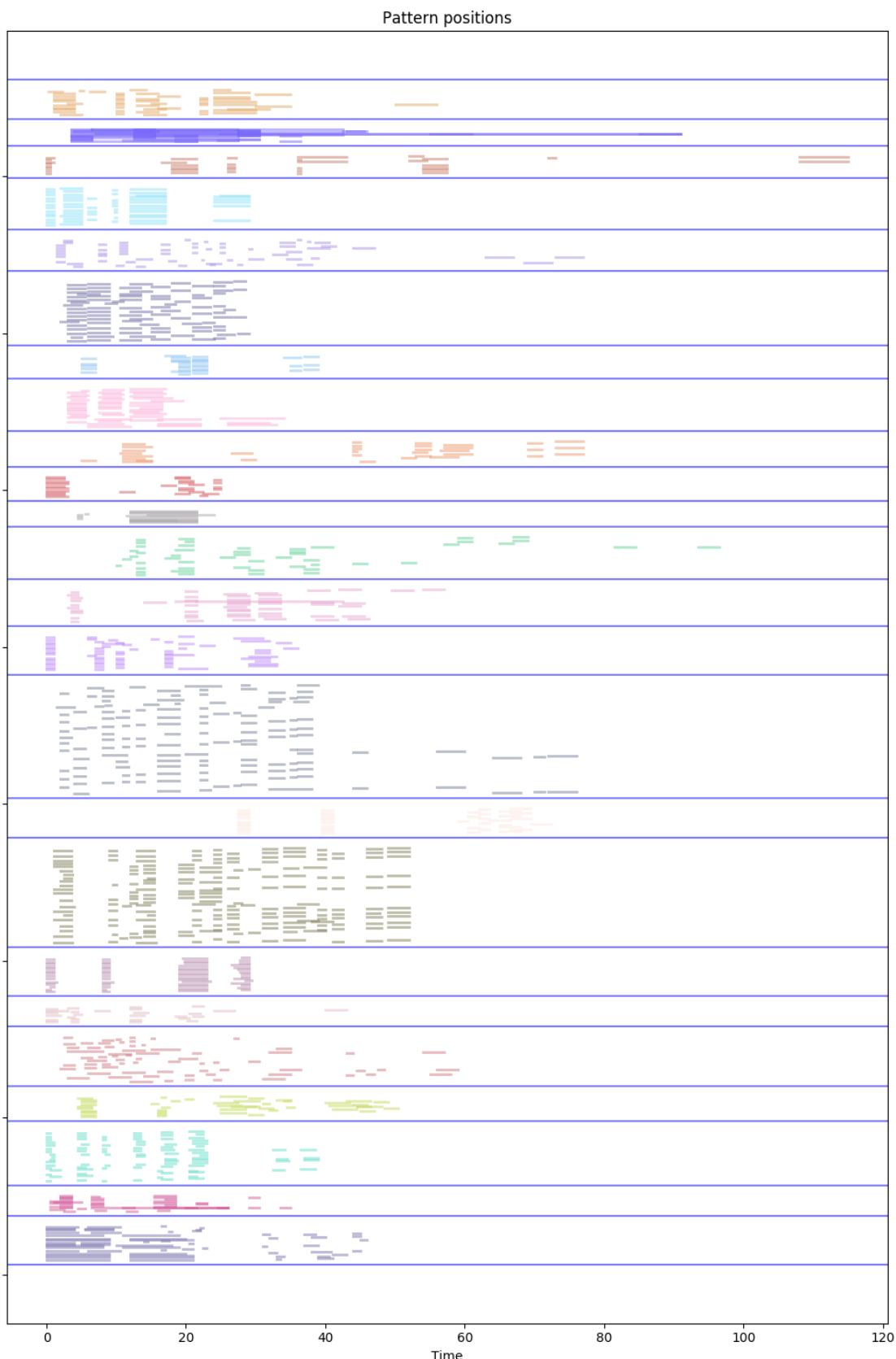


Figure 4.4: Visualisation of patterns in [MTC-ANN](#). Different colours and segments in the figure represent different tune families. Different vertical heights represent different pieces.



Figure 4.5: An augmented visualisation of Figure 4.4. In addition to the position, pitch and rhythm can also be plotted to an extent by plotting musical events using dots. The differences between pitches are reflected in the very slight differences in the vertical positions. Without an interactive zoom-in functionality, we cannot represent every detail precisely. This limitation of static visualisation points to a more interactive visualisation method in future work.

largest variances (spread) of the dataset. By examining the resulting principal components, we may gain insight into which features are of more significance in explaining the spread of the data points. By applying the principal components to perform a change of basis on the data points, we obtain a visualisation of the data points where the variances are clear. PCA has been employed and shown to be effective in a variety of MIR tasks, including (Van Balen, 2016).

To visualise patterns using their features and PCA, we first compute the features with a common feature extraction tool: the jSymbolic toolbox in the jMIR toolset (McKay, 2010).² jSymbolic takes MIDI files as input and computes 155 musically meaningful features in six categories: texture, rhythm, dynamics, pitch, melody and chords.³ Using the MTC-ANN dataset, we perform a feature selection step and retain 64 features as follows: (1) Eliminating the features which are constant across all patterns; (2) Eliminating the features which are irrelevant to the music content of time and pitch, such as the dynamics features. After this feature selection step, we perform PCA.

In Table 4.2.2, we report the prominent features and the weights in the first three PCA components. As the interpretation of the PCA visualisation does not require an understanding of the individual features and we will perform a more detailed feature analysis later in Section 4.2.4, we will not give a list of explanations of the features here, but some important features and their descriptions can be found in Table 4.6.

In Figure 4.6, we plot different groups of patterns extracted from MTC-ANN in a two-dimensional PCA embedding of the feature space. We make four cross-group comparisons to show typical cases of how musical patterns are distributed in the feature space spanned by the first two components of the PCA decomposition. The visualisation is generated by first computing the PCA embedding using the annotated patterns. Then, pattern features from different algorithms⁴ are projected onto this embedding. Finally, for baseline comparison, we extract random excerpts from the music, compute their features and add them to the PCA space. The comparison to the random baseline is of interest here because, otherwise, it would be more difficult to navigate the highly abstract PCA feature space. More specifically, the random excerpts are sampled using the following procedure:

²We will also use these pattern features in Section 4.2.4 for classification. PCA will also be applied as pre-processing step.

³There are multiple versions of jSymbolic. The newest version provides more features. The version we used to perform the experiments in this chapter is jSymbolic2.0.

⁴see Table 4.1 for a list of the algorithms with the changes of naming in the SIA family as before and VM1 to VM.

PC (Percentage of variance explained)	Features	Weight (Percentage)
PC1 (22.51)	Number of Strong Rhythmic Pulses Pitch Variety Number of Relatively Strong Rhythmic Pulses Number of Common Pitches Number of Moderate Rhythmic Pulses	5.18 5.15 5.07 5.07 5.07
PC2 (12.42)	Other Features Repeated Notes Relative Prevalence of Top Pitches Relative Prevalence of Top Pitch Classes Prevalence of Most Common Pitch Prevalence of Most Common Pitch Class	74.46 8.24 8.06 7.58 6.32 5.98
PC3 (8)	Other Features Combined Strength of Two Strongest Rhythmic Pulses Polyrhythms Rhythmic Variability Strongest Rhythmic Pulse Strength of Strongest Rhythmic Pulse Other Features	63.82 10.58 9.98 9.27 7.26 7.14 55.77

Table 4.2: The first three principal components of PCA and the weights of features the components consist of. We omit the rest of $64 - 3 = 61$ components since they do not contribute significantly ($< 7.5\%$) to the variance and, for visualisation purposes, we only use the first two components.

- For each annotated pattern in MTCANN, we find the corresponding song where the annotation appears.
- We then pick a random starting point and take an excerpt of the same length as the pattern to construct a candidate excerpt.
- Finally, we repeat the sampling procedure five times to control for anomalous results.

From the four snapshots we take from the musical pattern PCA feature space as shown in Figure 4.6, we make several observations:

- Annotated patterns and random excerpts have extensive areas of overlap, which makes it impossible to find a linear classifier that uses the first two principal components of the annotated pattern features, which in turn makes it nontrivial to differentiate the two groups of patterns as shown in the upper left subfigure.
- SIAR patterns exhibit a very different distribution from the annotated patterns and random excerpts as shown in the top right and left subfigures. Notice the annotated patterns concentrate at the top left corner in the top right subfigure. In this case, it is relatively easy to separate the long-tail area of the extracted patterns from the annotation area. By applying this observation and designing a filtering process, we could substantially improve the performance of the SIAR algorithm on MTCANN.
- The overlap between the annotated patterns and extracted patterns is small in the bottom left subfigure. A linear classifier can be devised to roughly separate the two groups of data using the first two principal dimensions of the annotated patterns. The extracted patterns of the SC algorithm have different features to the annotated patterns.
- In the bottom right subfigure, we show all the heterogeneous patterns extracted by algorithms, annotated by humans, and randomly sampled in the same PCA embedding. We can see that patterns extracted by algorithms of the same family, namely SIACP, SIACR, SIACF, and SIACFP tend to share the same long-tail property, and therefore their performance on MTCANN can be improved by an extra filtering step as described above.

The next steps following the observations

After examining two types of visualisations, we can take more concrete steps to further explore the pattern data. First, the observations made in visualising pattern locations hint at the possibility of combining similar patterns to devise an ensem-

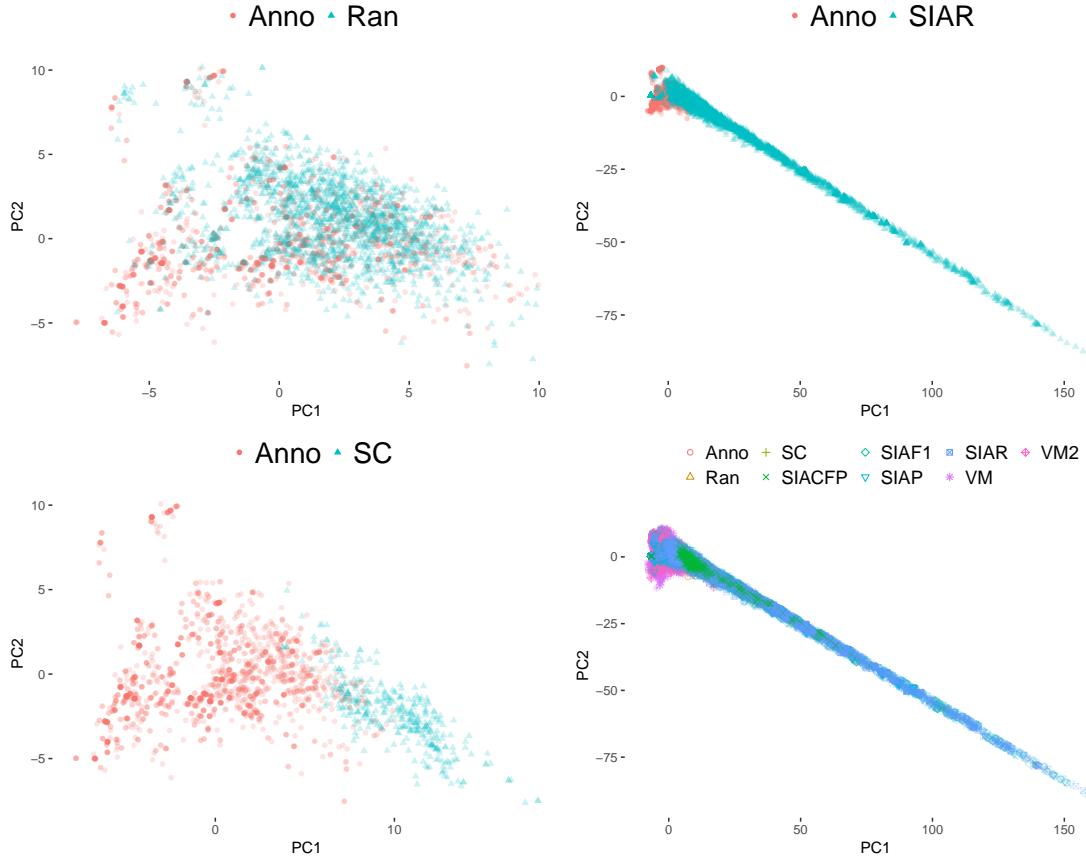


Figure 4.6: Visualisation of different groups of patterns using the space spanned by the first two principal components of the annotated pattern features in [MTC-ANN](#). The legend above each subfigure denotes the correspondence between colours and algorithmic patterns/annotated patterns/random excerpts. Notice that the figures in the left column show zoomed subregions of figures in the right column. (1) Upper left: random excerpts and annotated patterns. The overlap between the two groups is large, and it is nontrivial to separate them in this two-dimensional PCA embedding. (2) Upper right: [SIACR](#) patterns and the annotated patterns. [SIACR](#) patterns exhibit a long tail which is not shared by the annotated patterns. (3) Bottom left: [SYMCHM](#) patterns and the annotated patterns. The overlap of the data points is small, which makes it easier to separate the two groups in this embedding. (4) Bottom right: random excerpts, annotated patterns, and patterns from all algorithms. We can see that some of the algorithmically extracted patterns are very different from the annotated patterns, and the algorithms belonging to the same family exhibit similar long tails.

ble method. Second, from the PCA visualisation, going into the opposite direction, what seems promising is to use classification techniques to discriminate between different groups of patterns based on their features. Correspondingly, in Section 4.2.3 we will try to fuse the patterns from different algorithms, and in Section 4.2.4, we will try to classify between them.

4.2.3 Combining the algorithms

Data fusion

Ensemble methods, in which several algorithms are combined to achieve better performance, are becoming more common in various applications. Data fusion has also gained popularity recently, where data pre-processing is performed on different modalities, then concatenated to a new representation.

In addition to the trends and success stories, why should algorithms and data become combined in the first place? In (Mitchell, 2012), it was argued that data fusion could improve system performance in four ways: representation, certainty, accuracy, and completeness. The no-free-lunch theorem, which states that any two optimisation algorithms are equivalent when their performance is averaged across all possible problems, also provides a valid reason to do so.

Integrating different algorithms using data fusion has also been shown to be a successful approach to improving overall performance in dealing with ambiguous musical data, such as automatic chord estimation (Koops *et al.*, 2016). For musical pattern discovery, according to some similar pattern positions we saw in Section 4.2.2, there is hope of finding a consensus between various algorithms to achieve an overall better pattern discovery result. To this end, we devise Pattern Polling (PP), which takes the locations of discovered patterns from different algorithms as input, and outputs new locations of patterns based on this. We name the algorithm with "polling" because it involves measuring the level of consensus from an ensemble of algorithms.

Pattern Polling

We developed the PP algorithm based on the assumption that all pattern discovery algorithms aim to find passages containing a shared overall interest—the "patterns" in musical compositions. We view the output of each algorithm to be votes on whether a given time point participates in this pattern-salient part of the composition, i.e. is part of a musical pattern. We consider whether or not an algorithm recognises a pattern at any given time point to count as a "vote" on whether or not there is indeed a pattern present at that time offset. For the sake of convenience, we define the salience degree of a time point as the number of discovered patterns at this time offset. In essence, PP is a process in which each algorithm contributes to the salience degree of a time point based on their discovered patterns. The resulting polling curve is then taken as a base to detect new patterns with input from all algorithms.

Polling Curve

From the voting process described above, an example polling curve using several algorithms from the MIREX task is shown in Figure 4.7 (a). To elaborate on how we calculate the polling curve, we start with discretised time points $T = [0, 1, \dots, n]$ in the musical piece, with a resolution of one crotchet. If an algorithm finds a pattern occurrence at a given time point, we count that as one vote contributing to the pattern salience score at that time offset. We perform the same procedure for all pattern occurrences and sum the votes from all algorithms. In the end, we obtain the polling curve $P(t)$, which is a time series of voting counts at each time offset $t \in T$.

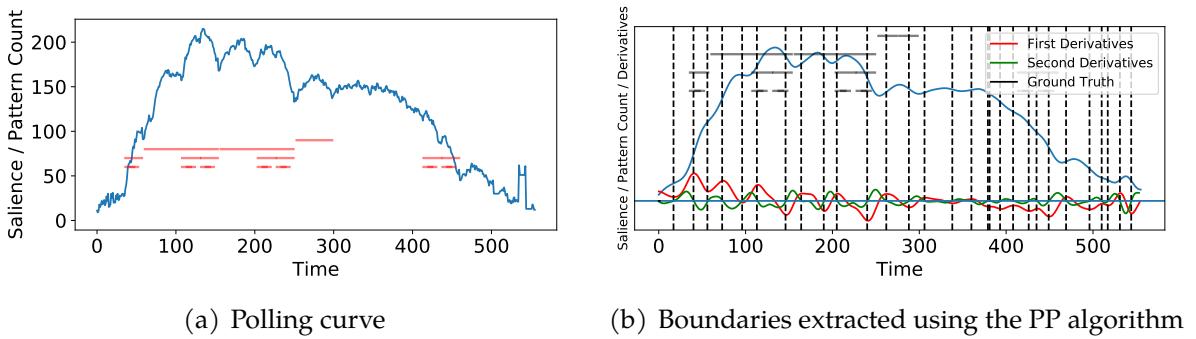


Figure 4.7: a. The polling curve of Chopin's Mazurka Op. 24 No. 4 using algorithms from the MIREX task. The horizontal bars show where the human-annotated patterns are present. The x-axis represents the time offset in crotchet units and the y-axis represents the salience value. We see promising correspondences between the polling curve and human annotations. b. Extracted pattern boundaries using PP indicated with dashed vertical lines. Many dashed lines are aligned with the boundaries of human annotations. We also plotted the polling curve, the ground truth human annotations, as well as both the first and second derivatives of the polling curve for reference.

Mathematically, we have:

$$P(t) = \sum_{A} \sum_{P} \sum_{O} I_O^{A,P}(t) \quad (4.1)$$

where A is the set of algorithms, P is the set of patterns, O is the set of occurrences, and $I_O^{A,P}(t)$ is the weighted indicator function of an occurrence of a pattern $p \in P$ in the output of an algorithm $a \in A$:

$$I_O^{A,P}(t) = \begin{cases} \omega_a & t \in o \subseteq p \subseteq a \\ 0 & t \notin o \subseteq p \subseteq a \end{cases} \quad (4.2)$$

where ω_a is the weight assigned to an algorithm a . Different values of ω_a may allow for the algorithms to be taken into account with different weights. In this

dissertation, we do not explore the possibility to weight the algorithms differently, and ω_a is always assigned to 1.

What polling curves represent and how to extract patterns from the curve

The polling curve uses the output from all individual pattern discovery algorithms. Given a music piece and a set of algorithms, it provides an estimate of how likely a region in music is to contain a pattern by taking into account all algorithmic output. The curve can be used for estimating where a pattern is more likely to be present in the musical piece. In Figure 4.7, we see a promising level of agreement between this estimated likelihood and human-annotated patterns.

To extract concrete patterns, we need to identify the beginnings and endings of patterns from the curve. The critical points of the curve can be helpful in this respect. Mathematically defined as the points at which the derivatives of the curve is equal to zero, critical points indicate the prominent changes in the shape of curves. Although the polling curve is a discretised curve, we can compute the discrete derivatives and corresponding critical points. These prominent shape changes then can be regarded as pattern boundaries. To distinguish between the prominent changes and those too small to be relevant, we first perform a smoothing step on the polling curve.

Smoothing

In our algorithm, we use the Savitzky-Golay filter (Schafer, 2011) for smoothing, which is a linear least-squares polynomial fitting filter. When we apply smoothing, we reduce the impact of small irrelevant changes in the curve at the cost of potentially losing valuable details. With different degrees of smoothing, we capture different levels of detail in the polling curve. With this in mind, we make our PP algorithm parametric on the degree of smoothness s .

Derivative

After smoothing, to find the prominent changes in the curve, we calculate the first and second discrete derivatives and take the critical points. More formally: let $P'(t) = P(t + 1) - P(t)$ and let $P''(t) = P'(t + 1) - P'(t)$, $t > 0, t \in T$. We are interested in zero-crossing points \bar{t} in $P'(t)$ and $P''(t)$ because each zero-crossing point \bar{t} represents a change of direction in the polling curve. For example, when $P'(t) < 0$ and $P'(t + 1) > 0$, we have a dipping point $P'(\bar{t}) = 0$ on the curve. More patterns are discovered by the algorithms starting from this point, so it is likely to be the start of a pattern.

One question remains as to how strong the dipping, tipping, concave, and convex positions in the curve should be if we are to pick them as boundaries. Here we

Algorithm	Precision	Recall	F1
ME	(0.125, 0.086)	(0.184, 0.077)	(0.149, 0.083)
SC	(0.396, 0.022)	(0.419, 0.068)	(0.402, 0.046)
OL1	(0.420, 0.038)	(0.565, 0.044)	(0.462, 0.023)
OL2	(0.422, 0.061)	(0.565, 0.044)	(0.483, 0.054)
SIAF1	(0.139, 0.049)	(0.670, 0.005)	(0.228, 0.041)
SIAR	(0.213, 0.039)	(0.427, 0.000)	(0.279, 0.021)
SIAP	(0.117, 0.043)	(0.596, 0.008)	(0.195, 0.037)
VM1	(0.137, 0.035)	(1.0, 0.0)	(0.240, 0.029)
VM2	(0.206, 0.073)	(0.543, 0.024)	(0.296, 0.060)
SIACFP	(0.819, 0.030)	(0.82, 0.064)	(0.815, 0.046)
PP-P	0.478	0.206	0.249
PP-R	0.228	0.867	0.35
PP-F1	0.248	0.738	0.360

Table 4.3: A table of the (mean, variance) of the precision, recall, and F_1 scores of the pattern boundaries of different algorithms. The PP-P, PP-R and PP-F1 are obtained using a 3-fold cross-validation training process optimising precision, recall, and F_1 score. Because we only have one piece in the test set, there is no variance value. The best results from individual algorithms and PP are shown in bold.

introduce a second parameter of the PP algorithm: a threshold on the steepness of the zero-crossing points λ . With different values of λ , we create a set of boundaries that consists of time offsets at which zero-crossings occur.

In Figure 4.7, we show an example of the extracted boundaries. We notice that some boundaries line up well with human-annotated pattern boundaries.

Evaluation metrics

To evaluate the accuracy of the extracted pattern boundaries, we choose to use the ground truth of human-annotated patterns. We compare the computed boundaries with the beginnings and endings of patterns marked by humans with standard evaluation metrics of precision, recall, and F_1 score. Following these standard evaluation metrics, the metrics we also used in Chapter 3, we calculate the [precision](#), [recall](#), and [\$F_1\$ score](#) with a degree of fuzziness, i.e. we attempt to match the boundaries with a tolerance of one crotchet note length as this is the degree of discretisation we used for creating the polling curve.

Results

Using the setup above, we extract patterns using the PP algorithm in a subset of JKUPDD¹. The original JKUPDD dataset contains five pieces, and we take three pieces in the monophonic format: Chopin's Mazurka Op.24 No.4, Mozart's Piano

Sonata K.282, 2nd movement, and Beethoven’s Piano Sonata Op.2 No.1, 3rd movement. The other two pieces contain a concatenation of voices from the polyphonic version, which violates the assumptions of geometric algorithms—these algorithms cannot treat polyphonic music as a concatenation of voices—and are therefore excluded.

The results are shown in Table 4.3. By calculating the mean values of all algorithms, we can see that the best F_1 score of PP 0.360 is slightly better than the mean of the F_1 scores of individual algorithms 0.3549. When we look at the individual algorithms, the best F_1 score of PP ranks fifth out of ten. The SIACFP algorithm performs the best overall. Although we also observe that PP performs slightly better than the average of the individual algorithms, we cannot yet conclude that this fusion method improves the accuracy of pattern discovery significantly.

Why is there no significant improvement?

From the results, we identify some potential reasons as to why PP does not outperform individual algorithms. Firstly, the available dataset is small, and the human-annotated patterns are sparse, which is problematic for training the parameters in PP. Secondly, the algorithms disagree with each other on pattern length, pattern overlap, and the number of patterns, which may be caused by a variety of different factors, including: the inherent ambiguity of music and pattern perception; the lack of a unified goal; the different target applications of the musical pattern discovery algorithms; or a combination of all of these factors. In the end, although we observed a promising level of consensus between algorithms in Figure 4.2 and Figure 4.7, Table 4.3 reveals that this is not yet sufficient to extract patterns that substantially agree with the human-annotated patterns.

From location to content, from combination to discrimination

So far, in this PP experiment, we have looked at the locations of patterns yielded by different pattern extraction algorithms and explored some of the possible ways in which they might be combined. To further investigate the musical events in these patterns, we move on to the pattern features and use them to distinguish between computed patterns and human annotations.

4.2.4 Classification

In the previous section, we compared algorithms using the locations of patterns. In Section 4.2.2, with PCA, we have extracted features from musical patterns and visualised them in the feature space using MTC-ANN¹. In this section, using the same

feature data, we perform Comparative Classification (CC) between the features of algorithmically extracted patterns, human annotations, and random excerpts, to see whether **SotA** classifiers can separate the patterns identified by different agents (algorithms, humans, and randomness) consistently.

We perform CC on two subtasks for two different levels of comparisons. The first classification task is to classify patterns into three groups based on how they were extracted: algorithmically, manually, or randomly. In the second task, we perform a finer level of classification on the algorithmic group from the first task by classifying patterns based on the algorithms that extracted them. In both tasks, we expect classifiers to help us discriminate between groups of patterns based on their musical features. We then explain the classification results using feature importance analysis.

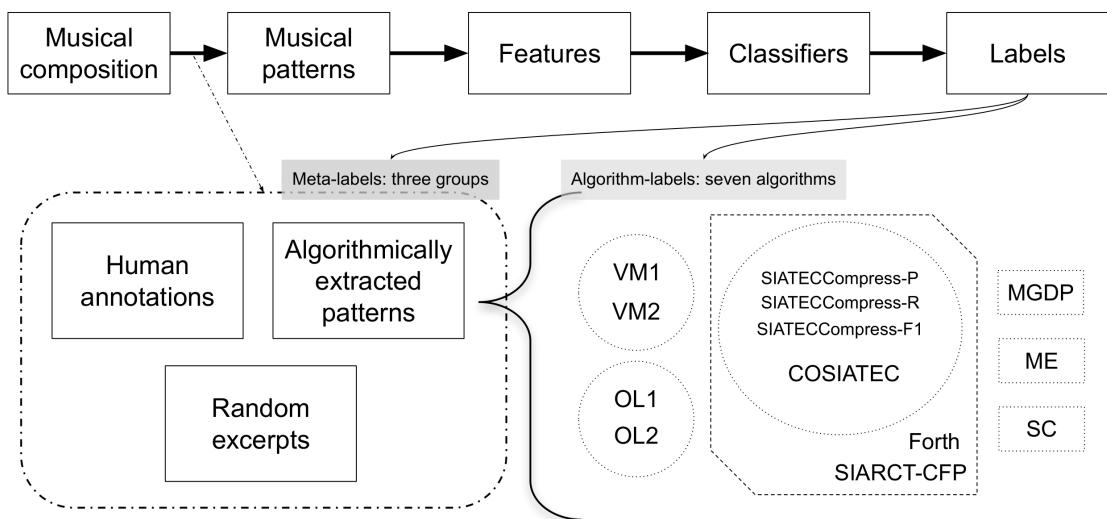


Figure 4.8: Pipeline of CC (the chain on top) and the two classification tasks using two types of labels. The meta-label classification experiment tries to use classifiers to distinguish between three groups of patterns from different origins. The algorithm-label classification experiment is performed at a finer level: we look into the algorithmically extracted patterns specifically and distinguish between seven subgroups of patterns. The seven algorithms we consider are summarised in Table 4.1. The geometric methods are in the dashed line square with cut corners. Other algorithms are also listed to be considered in the next section.

Supervised classification methods have been used extensively in MIR tasks such as genre classification and the classification of corpora from different geographic origins. Leveraging the discriminative power from classification algorithms for distinguishing the output from other algorithms is related to adversarial machine learning, which is gaining popularity (da Fontoura Costa & Cesar Jr, 2009; Patel & Barkovich, 2002). To the best of our knowledge, the pipeline we propose in Figure 4.8 has not yet been used to evaluate symbolic musical pattern discovery algorithms.

Note that extracting features from patterns themselves is equivalent to analysing the patterns out-of-context. Analysing musical passages in isolation from their context might be limited in generalisability because a pattern in one musical piece might be insignificant in another piece depending on what surrounds it. Nevertheless, certain arrangements of musical events and their features could be important in signifying the differences between algorithmically extracted patterns and human annotations.

Classifiers

To prevent the results being classifier-specific, we use a mixture of simple and more sophisticated, linear and non-linear classifiers to perform the classification tasks. We also use standard machine learning techniques to train and test classifiers: scaling and centring preprocessing steps are first performed on all the features. Furthermore, we create another set of features by applying PCA to the jSymbolic features to see whether the PCA features will perform better than the original ones. Additionally, to avoid overfitting, for all experiments, we use a 10-fold *cross-validation* 3-times repetition scheme. The parameter search for each classifier are performed separately on each fold. The six statistical classifiers we use are:

- **Gradient Boosting Machine (GBM)** (Friedman, 2001) produces a prediction model consisting of an ensemble of decision trees. The parameters we search through are the learning rate, complexity of trees, minimum number of samples to commence splitting, and the number of iterations.
- **Linear Vector Quantisation (LVQ)** (Kohonen, 1990) applies a winner-takes-all Hebbian learning-based approach. We search through two parameters in this classifier: codebook size and number of prototypes.
- **Linear Discriminant Analysis (LDA)** (Ripley, 2007) produces a linear classifier which finds a linear combination of features that best separates different classes in datasets. This classifier is not parametric.
- **Naive Bayes (NB)** (Ng & Jordan, 2002) computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. Three parameters are tuned for this classifier: Laplace smoothing, kernel bandwidth, and distribution type.
- **Random Forest (RF)** (Breiman, 2001) operates by constructing a multitude of decision trees. The parameter we consider is the number of variables per level.
- **Support Vector Machine (SVM)** (Suykens & Vandewalle, 1999) calculates a map from data to a new representation so that the data points of the different categories are separated by as large a gap as possible. We use the radial basis

function kernel and consider two parameters: smoothing factor and weight of training examples.

Classification results

We mainly use accuracy and the variance of accuracy as our measures of the performance of the classifiers. To further interpret the results of the classification task, we compute confusion matrices and feature importance measures.

Model metrics

Figure 4.9 shows the accuracy and variance of different classifiers in the two classification tasks. We use two groups of features, the raw features and features after the PCA step. The baseline accuracy is $\frac{1}{\#\text{group}} = \frac{1}{3} = \sim 33\%$ for the first task and $\frac{1}{\#\text{group}} = \frac{1}{7} = \sim 14\%$ for the second. We balanced the number of patterns in each group to make them the same, for the first task = 1657, and for the second task = 355.

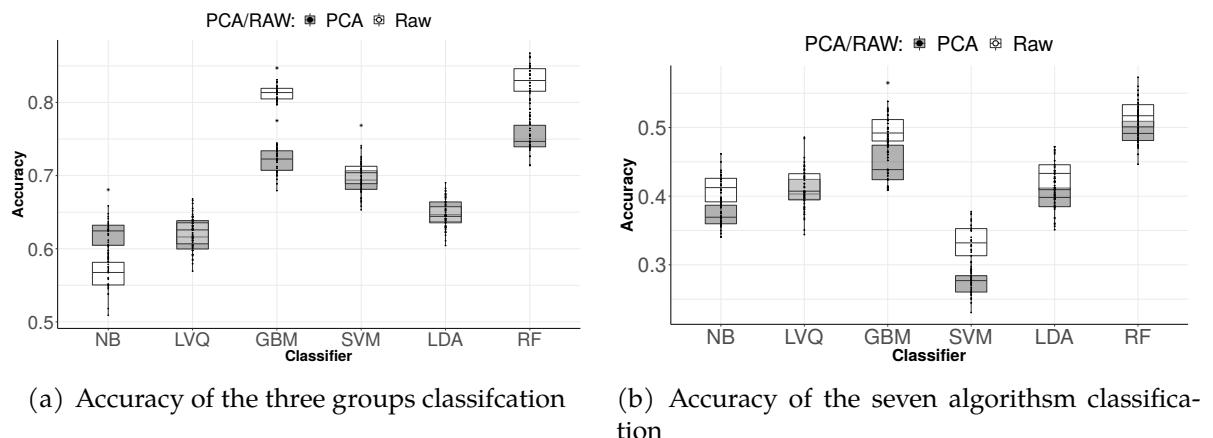


Figure 4.9: Accuracy values for classifiers in thirty experiments (10-fold cross-validation repeated three times) using six classifiers with jsymbolic2.2 features and after PCA preprocessing.

We see that all classifiers give a result higher than the baseline accuracy in both tasks. PCA improves the performance of the classifier **NB** in the first task. For **LVQ**, **SVM**, and **LDA**, using PCA or raw input does not make a significant difference. The performance of other classifiers is worse when using the PCA input. The finding that PCA has different influences on the performance of classifiers may be due to the fact that each classifier uses different internal feature transformation mechanisms. Overall, the random forest classifier gives the best results for both tasks with the raw feature input and the parameter `#variables = 32`.

The fact that classifiers can differentiate between groups of patterns with above-baseline accuracy values has a few implications. In the first task, it implies that algorithmically extracted patterns possess different properties than human-annotated patterns, which suggests that extra consideration of pattern features would be beneficial when trying to discover patterns automatically. It also shows that the algorithmically extracted patterns have different traits than random excerpts, which means that these patterns are not equivalent to randomly sampled excerpts, and are therefore potentially more useful for some applications. Lastly, it shows a difference between human-annotated patterns and randomness despite subjectivity being involved in the annotation process, which is consistent with the carefully designed annotation collection process (van Kranenburg *et al.*, 2016) and previous findings that the annotations are useful for classifying tune families (Boot *et al.*, 2016).

In the second task, the above-baseline accuracy shows distinguishability in the patterns extracted by the algorithms, which suggests disagreement between different algorithms, reinforcing our conclusions from the first experiment. To further investigate differences between groups, we now analyse the confusion matrices.

Confusion matrices

Table 4.4 and Table 4.5 give the confusion matrix results calculated from the classifier with the best classification results: Random Forest. We perform the repeated cross-validation experiment ten times and take the mean and variance of the resulting ten confusion matrices. The results show us that different groups of patterns are separable and dissimilar to one another according to the random forest classifier.

To read the table, we first notice that the sum of each column is roughly 500, which is the size of our test data. The row sums do not have this constraint because there are no restrictions on the group sizes as determined by the classifier. For interpreting the entries in the table, we take as an example the number 29.5 in the top right corner of the table. This number is the mean count of patterns classified as algorithmically extracted patterns but are actually human annotations. Table 4.5 is formatted similarly, with a different column sum because of the different test data size, and with algorithm names instead of group names.

In the first task, we see that the classifier can differentiate between the three groups with few instances of incorrect classification. This classification result is not the most desirable for the pattern discovery algorithms. If we had seen that the classifiers could not differentiate between the algorithmically extracted pattern group and the human-annotated pattern group, this would suggest a level of consensus between

Original → Classified ↓	Alg	Ran	Anno
Alg	406 (± 13.5)	32.3 (± 7.2)	29.5 (± 8.3)
Ran	54.1 (± 8.9)	402 (± 15.9)	65 (± 14.1)
Anno	37 (± 8.4)	62.8 (± 11.3)	402.6 (± 14.2)

Table 4.4: Confusion matrix results from the ternary classification experiment using the Random Forest classifier: mean and variance (in parenthesis) of ten classification experiments. The row names indicate that the patterns are classified into the group with this name by the classifier; the column names indicate the patterns are originally from the group with this name. Three groups of data are classified with high accuracy and significant p-values $\ll 0.05$.

algorithms and humans. In other words, if the count is larger at the last line of the first column in the confusion matrix, it would indicate that the patterns discovered by algorithms are harder to distinguish from the human annotations, which is more desirable if the algorithms would like to imitate the pattern discovery behaviours of human annotators. With the current results, however, we come to the conclusion that the algorithmically extracted patterns, annotated patterns, and random excerpts possess their own traits and are not similar enough for the classifiers to fail. This is in accordance with the first experiment and previous research, which shows that the algorithmically extracted patterns are not yet indistinguishable from the human annotations (Boot *et al.*, 2016; Ren *et al.*, 2017). Despite this, we at least establish that neither annotated patterns nor extracted patterns are equivalent to random data.

In the second task, the classifier can also differentiate patterns from different algorithms to a certain extent. The goal of this task is to examine the individual pattern discovery algorithms. Similarly to the analysis in the first subtask, if we can distinguish between different algorithms perfectly, this would indicate that the discovered patterns are very different, despite the algorithms having the same objective of "pattern discovery"; and if the algorithms are in agreement with each other, we would expect the classifier to fail and the values in the confusion matrix to be more uniformly distributed. We see in Table 4.5, however, this is often not the case. The exceptions here are the patterns extracted by algorithms from the same family (SIACF, SIACP, SIACR, for example): the classifier performs worse in distinguishing within the same family. This is an indication that we are not overfitting the classifier. More importantly, this classification result agrees with the first data fusion experiment in that the output from algorithms has a high degree of disagreement, especially when the algorithms come from different families.

Original → Classified ↓	SIACF1	SIACP	SIACR	VM2	VM1	SC	CFP
SIACF1	22.6(± 9.7)	18.6(± 9.9)	23.6(± 12.8)	2.5(± 2.3)	0.4(± 0.8)	1.4(± 1.5)	8.4(± 4.7)
SIACP	12.8(± 6.8)	34.4(± 6.6)	15(± 5.9)	0.5(± 0.9)	1.3(± 1.8)	1(± 1.9)	5.1(± 2.5)
SIACR	30.6(± 12.1)	23(± 8.7)	31.8(± 9)	1.8(± 2.1)	1(± 1.3)	1.3(± 1.9)	10.6(± 5.6)
VM2	8.2(± 5.1)	11.1(± 4.2)	9(± 4.4)	63.1(± 7.2)	21.6(± 7.8)	0.3(± 0.6)	3.8(± 2.5)
VM1	0.9(± 1.4)	5.5(± 2.9)	1.9(± 2.2)	22.5(± 6.9)	78.5(± 8.8)	0(± 0)	0.3(± 0.9)
SC	16.2(± 6.5)	6.8(± 4.2)	10.3(± 4.8)	7.3(± 3.9)	0.2(± 0.6)	93.8(± 4.9)	17.1(± 4.1)
CFP	15.2(± 5.9)	7.2(± 4)	14.9(± 5.6)	8.7(± 4.6)	3.6(± 3.1)	8.8(± 4.5)	61.1(± 9.2)

Table 4.5: Confusion matrix results classifying the algorithmic output using the Random Forest classifier. The columns and rows follow the same format as in Table 4.4.

Feature importance

Figure 4.10 and 4.11 show the individual importance value of each feature used in the random forest algorithm (the best classifier in the two classification tasks), computed by using the Boruta algorithm (Kursa, Rudnicki, *et al.*, 2010). The Boruta algorithm randomly duplicates and shuffles the values in the original features as extracted by jSymbolic. The algorithm then combines some of these randomised features with the originals in order to calculate and compare the significance of different combinations of features. At the end of this process, we obtain an importance value for each feature—the Gini impurity importance value (Louppe *et al.*, 2013).

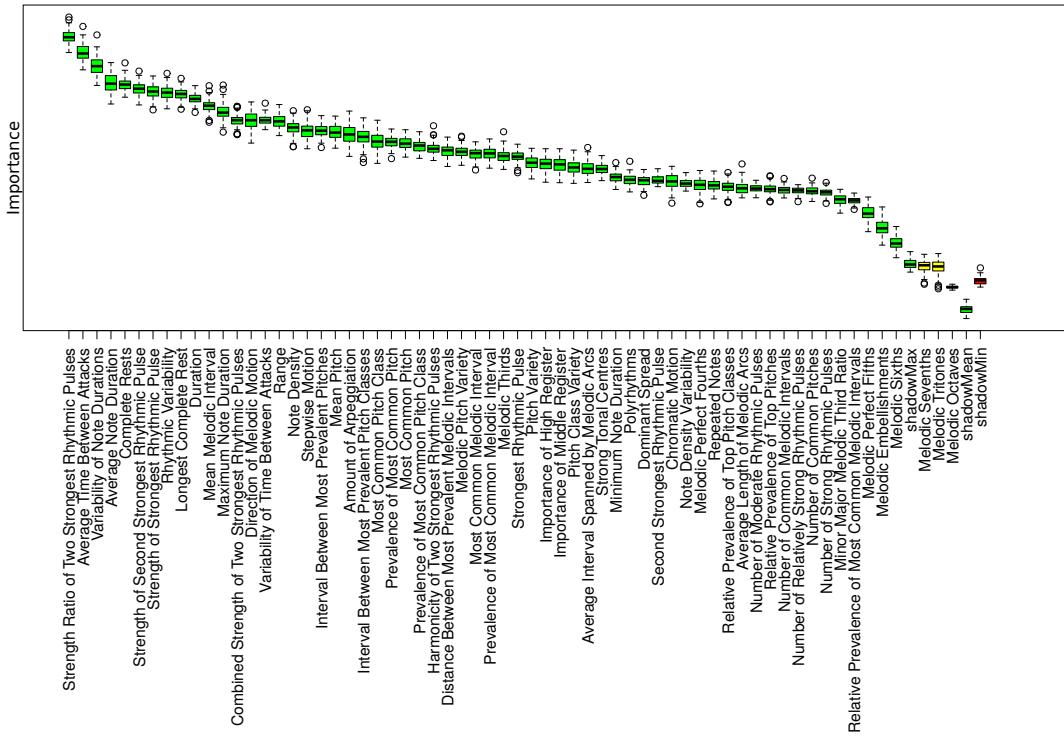


Figure 4.10: Feature importance for differentiating between annotation, algorithmic output, and random excerpts. The boxplot depicts the mean and variance (interquartile ranges) of the feature importance values (Kursa, Rudnicki, *et al.*, 2010). The features are ranked by their importance. We omit the y-axis label because the absolute importance values are not relevant for our analysis. The colour green indicates features that are more important than the randomised features and are therefore confirmed to be significant; blue entries show the performance of the random features; red and yellow indicate unimportant and tentative features, respectively.

Here, in Table 4.6, we summarise the top 10 important features in the classification for Figure 4.11 (the top part of the table) and Figure 4.10 (the bottom part of the table). The descriptions are taken from the most relevant part of the developers'

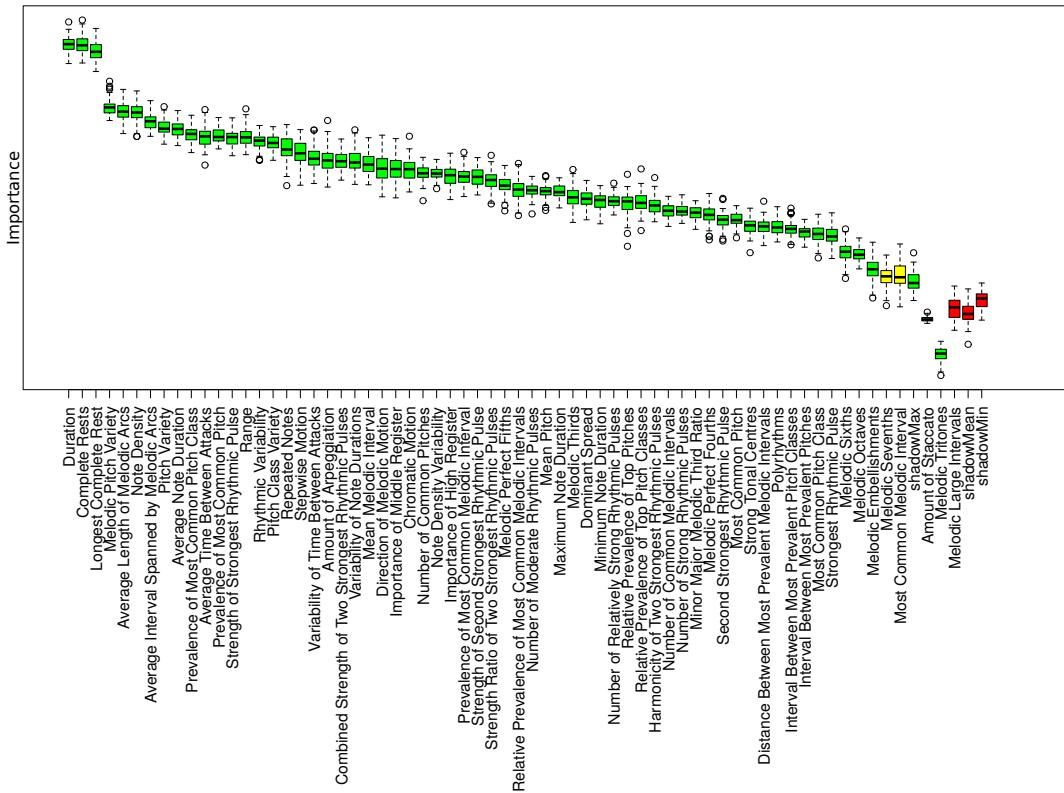


Figure 4.11: Feature importance for differentiating between output from different algorithms in the same format as Figure 4.10.

documentation of jSymbolic. Several features in Table 4.6 use another feature: Beat Histogram. For reference, this feature was defined as

"A feature vector consisting of the bin magnitudes of the beat histogram described above. The first 40 bins are not included in this feature vector... Each bin corresponds to a different beats per minute periodicity, with tempo increasing with the bin index. The magnitude of each bin is proportional to the cumulative loudness (MIDI velocity) of the notes that occur at that bin's rhythmic periodicity. The histogram is normalized."

Another convention adopted when calculating rests in the feature is that "Non-pitched (MIDI channel 10) notes are not considered in this calculation. Rests shorter than 0.1 of a quarter note are ignored".

Another well-known feature extraction package, the FANTASTIC toolbox (Müllensiefen, 2009) is not used because its minimum supported input length excludes valuable annotated patterns. jSymbolic was also not specifically designed for short excerpts, and many human-annotated and algorithmically extracted patterns are indeed short in length. Nevertheless, jSymbolic computes features in the same way for

all excerpts; it does not, therefore, undermine the validity of the classification experiments.

Implications

Although there are only 23 rhythmic features out of 63 in total in the jSymbolic features we used, the features ranked highest are rhythmic in nature. This high ranking of rhythmic features suggests that these features were more important than other features in constructing the random forest classifiers, which hints at potential improvements that could be implemented for current existing pattern discovery algorithms. String-based and data mining algorithms translate pitch and duration pairs into a list of symbols and therefore do not take into account metric structures imposed by musical punctuation such as bar lines. This is not uncommon as other algorithms also seldom explicitly consider metric features in patterns. In particular, the [SYMCHM](#) algorithm considers pitch aspects only, which explains its limited overlap with human-annotated patterns in the [PCA](#) visualisation. Generally, the feature importance values we obtained suggest that in designing and evaluating pattern discovery algorithms, at least for the [MTC-ANN](#) dataset, we should take metric structures into consideration as well as repetition and pitch related features in patterns.

Our findings suggest that other jSymbolic features are almost all important for the classifiers, with the exception of three, which performed worse or at the same level as randomised features. The Boruta algorithm categorises Melodic Octaves feature as unimportant and the Melodic Sevenths and Melodic Tritones as tentative features. They are indeed nonessential features because these musical intervals rarely appear in the MTCANN dataset.

4.2.5 Synthetic structure and data

In previous sections, to understand where and how the algorithms disagree with each other, we computationally examined the musical patterns by visualising, combining, and classifying them. Through this algorithm-algorithm and algorithm-human comparison, we see that a diverse range of patterns can be extracted by algorithms and annotated by humans. While visualisation, PP, and CC help us understand how the patterns resemble and differ from one another, we still lack an intuitive understanding of how algorithms would perform in the presence of new musical corpora.

Feature name	Feature explanation
Strength Ratio of Two Strongest Rhythmic Pulses	Magnitude of the beat histogram peak with the highest magnitude divided by the magnitude of the second highest magnitude.
Average Time Between Attacks	Average time (in seconds) between Note On events (regardless of MIDI channel).
Variability of Note Durations	Standard deviation of note durations (in seconds).
Average Note Duration	Average duration of notes (in seconds).
Complete Rests	Fraction of the music during which no pitched notes are sounding on any MIDI channel.
Strength of Second Strongest Rhythmic Pulse	Magnitude of the beat histogram peak with the second highest magnitude.
Strength of Strongest Rhythmic Pulse	Magnitude of the beat histogram bin with the highest magnitude.
Rhythmic Variability	Standard deviation of the tempo-standardized beat histogram bin magnitudes.
Longest Complete Rest	Longest amount of uninterrupted time (expressed as a fraction of the duration of a quarter note) in which no pitched notes are sounding.
Duration	Total duration (in seconds) of the piece (pattern in our case).
Melodic Pitch Variety	Average number of notes that go by in a MIDI channel before a note's pitch is repeated (including the repeated note itself).
Average Length of Melodic Arcs	Average number of notes that separate melodic peaks and troughs.
Note Density	Average number of notes per second.
Average Interval Spanned by Melodic Arcs	Average melodic interval (in semitones) separating the top note of melodic peaks and the bottom note of adjacent melodic troughs.
Pitch Variety	Average number of notes that go by in a MIDI channel before a note's pitch is repeated (including the repeated note itself)
Prevalence of Most Common Pitch Class	Fraction of notes that correspond to the most common pitch class.

Table 4.6: Table of feature description. The features included are the top 10 features in Figure 4.11 (the top part of the table) and Figure 4.10 (the bottom part of the table). Only the most relevant description from the documentation of jSymbolic was included.

Given the complexity of musical corpora, we propose the use of synthetic data in conducting an initial assessment of the performance of musical pattern discovery algorithms. For example, one of the simplest and yet most fundamental questions to ask is whether pattern discovery algorithms are capable of identifying patterns in sequences such as "Pattern1" + "Pattern2" + "Random Excerpts" + "Pattern1". By artificially constructing a concatenation of musical patterns and random sequences of notes and rests, we can compare the performance of different algorithms at extracting patterns from sequences such as this—random sequences with patterns artificially inserted throughout, giving a controlled amount of regularity. This is a method that has been widely used in other areas such as generic pattern mining and time series analysis (Bertens *et al.*, 2016; Chiu *et al.*, 2003). To the best of our knowledge, it has not yet been used extensively to compare musical pattern discovery algorithms.

In this section, we examine seven musical pattern discovery algorithms using synthetic data. Because not all algorithms are open-sourced, we are only able to obtain the patterns extracted by algorithms on certain datasets. Refer to Table 4.1 for the algorithms we added and removed from previous experiments in this section. We will now first describe how we created the synthetic data and then show an example piece from which different algorithms extract different patterns.

To create the synthetic data, we randomly concatenated two predefined patterns with random excerpts. Given that we have planted the patterns artificially, we are able to compare the output of the algorithms to what is ostensibly a "ground truth". The details of the predefined patterns are given next.

One of the predefined patterns is a consecutive repetition of the notes C and G# in the fourth octave. We refer to this pattern as P_1 , the repeated interval pattern. The other pattern we use is a C major scale, also known as the C Ionian scale in a modal context. We refer to this pattern as P_2 , the scale pattern. Both patterns are twenty notes long. We chose these two patterns because they are of different kinds: P_1 contains many local repetitions, while P_2 is an example of a global pattern with a longer span. The excerpts of the patterns can be seen in Figure 4.12.

We sample the random excerpts, R , with the same note range as the scale pattern. We sample rests as well as notes. The lengths of the random excerpt are not fixed, but we constrain the total length of the random excerpts to be less than 50% of the total length of the synthetic piece.

Although this is a simple set-up, it follows two rationales that we have in mind. First, we would like to reduce the challenge posed by rhythms and put more focus on investigating how pattern discovery algorithms detect pitch related patterns. There-

fore every note and rest in R , P_1 , and P_2 has the note length of a crotchet. Second, the repeated interval patterns such as P_1 can be used to test whether the algorithms can retrieve local repetitions of intervals, scale patterns such as P_2 can be used to test whether the algorithms can identify global repetitions of an organised sequence of notes, and random excerpts such as R can be used to test whether the algorithms will recognise patterns from randomness.

We present one piece from our set of synthetic pieces to illustrate the differences in the output produced by the algorithms. The point-set visualisation of the piece can be seen in Figure 4.13, 4.14, and 4.15. Figure 4.12 visualises the presence of discovered patterns along the time axis of the synthetic piece, as introduced in Section 4.2.2. We further discuss our observations with respect to individual algorithms below.

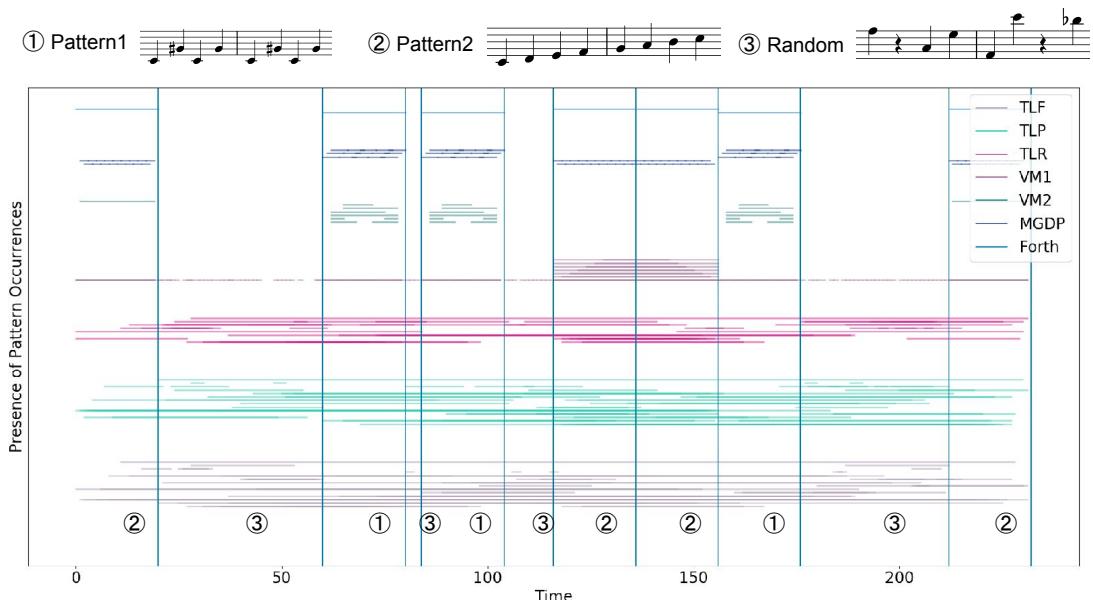


Figure 4.12: Visualisation of the presence of patterns using synthetic data.

VM

We observe that VM1 consistently finds P_2 , and VM2 likewise for P_1 . However, the algorithms discover multiple subpatterns in the regions where P_1 and P_2 are present. Regarding the PP method introduced in Section 4.2.3, if we summarise algorithmic output by summing the counts of the discovered patterns, the resulting polling curve will give a correct indication of their presence. This tells us that the algorithms could benefit from a combined approach, though only in specific circumstances.

MGDP

Patterns discovered by the MGDP algorithm are short patterns. The parameters used were $\alpha = 0.1$, $\text{Min(support)} = 20$, $\text{Viewpoint} = \text{Interval}$. The discovered interval patterns are $\{8, -8, 8\}, \{-8, 8, -8\}, \{2, 2, 1\}, \{1, 2, 2\}, \{2, 1, 2\}$, which correspond to parts of P_1 and P_2 . We can see from Figure 4.12 that the trigram (three-component) interval patterns cover P_1 and P_2 completely, though with multiple overlapping occurrences.

Forth

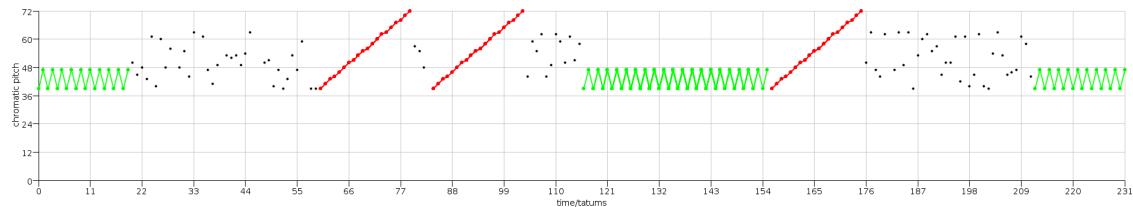


Figure 4.13: Patterns retrieved by the Forth algorithm

The Forth algorithm successfully retrieved all the planted patterns in the example piece. As we can see in Figure 4.12, by comparison with the ground truth (numbered (1)-(3)), the two planted patterns and their occurrences are all retrieved. In Figure 4.13, we can see in more detail that the algorithms find both the repeated intervals and the scales.

SIARCT

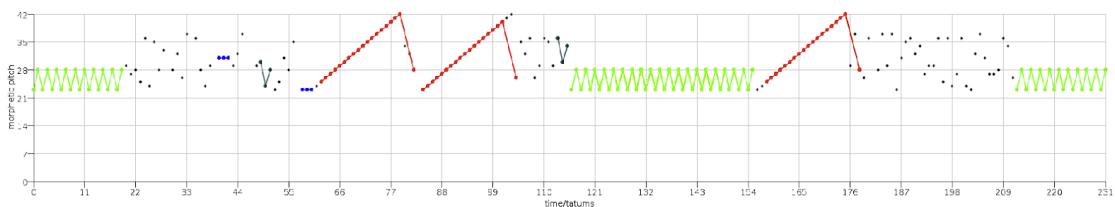


Figure 4.14: Patterns retrieved by the SIACFP algorithm.

As shown in Figure 4.14, while patterns retrieved by SIARCT are indeed repeated patterns in the piece, they tend to deviate from P_1 and P_2 : the scale patterns are corrupted by adjacent notes in the random sequences, and the algorithm finds a small number of patterns in the random pitch sequences. Nevertheless, the approximate structure of the piece is retrieved.

SIACPRF

Without parameter optimisation and applying out-of-the-box SIACR, SIACP, and SIACF algorithms, we find many patterns in addition to the ones that are injected into the piece in Figure 4.12. The output does not contain occurrences that are confined to within the boundaries of the planted patterns. Not only this, but most of the extracted patterns cover a span of time in the piece that is much longer than the planted patterns.

Further observations

Patterns with long spans

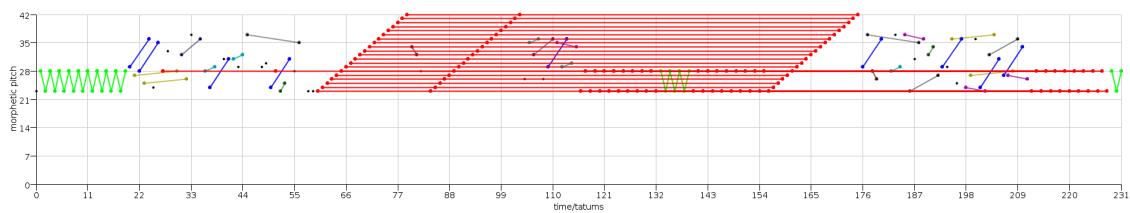


Figure 4.15: Patterns retrieved by the COSIATEC algorithm.

Figure 4.15 shows long-span patterns which, with a concrete view of the example piece, can be seen to be unintuitive but not unreasonable: the injected scale P_2 repeats across the piece, but the elements within the scales also repeat individually. This repetition is less intuitive because of limited human memory and the Gestalt principle of proximity in grouping notes (Lerdahl & Jackendoff, 1985; Snyder & Snyder, 2000). It is more plausible for an audience to identify the scale as a musical pattern rather than each individual note over a long time span. There is a duality between the locally bounded repetition (where the scale repeats as a whole) and globally identifiable repetition (notes in scale repeat separately). Depending on the subsequent applications of the extracted patterns, the algorithm should give more consideration to balancing or filtering out patterns of the desired kind. For example, if the patterns are to be used to assist with musical motif analysis, the local patterns are more valuable; if our goal is to uncover hidden structures in music, global patterns could open up more possibilities and insights.

Comparison with ground truth

As we have discussed regarding long-span patterns, it can be difficult to automatically evaluate the patterns because multiple patterns can be extracted from the same constructed piece, and it is not fair to determine the correctness of the discovered

patterns without application scenarios in mind. In addition, we must also consider the issue of intentionality in the use and recognition of planted patterns. On the one hand, unintended patterns can emerge from the concatenation of patterns and randomness. In other words, it is possible to inadvertently introduce inexact and exact repetitive patterns in addition to those we planted deliberately. On the other hand, the two patterns we employed are distinctive enough that it is likely that a human annotator would be able to find the exact boundaries of both. Nevertheless, there are known cognitive phenomena such as apophenia (Steyerl, 2016), patternicity (Shermer, 2008), and hyperactive agency detection (Valdesolo & Graham, 2014). If there is a tendency to find meaningful patterns in meaningless noise (Fyfe *et al.*, 2008; Shermer, 2008) in human subjective experience, the question of how one should evaluate the algorithmic output in relation to the ground truth human annotations remains a complex one. Therefore, despite the differences we observe in all three experiments, we cannot repudiate the value of the discovered patterns from algorithms.

Rhythmic features eliminated

Notice that, in the example above, we reduced the variance in rhythmic features by using the same duration for each musical event. Rhythmic features, being the most divergence-inducing factor in CC, should give the algorithms more tendency to conform, once we control their variance by artificially increasing their regularity. Adding in more rhythmic variation could lead to greater insight into how the algorithms handle the interplay between pitch and rhythm. We defer a more systematic investigation on this topic to future work. The minimal example we provided, despite being restricted to pitch, informs us of crucial aspects of the algorithms with respect to the pitch dimension. This is another advantage of using synthetic data—it gives us a higher level of control in investigating musical features.

Implications for the human-algorithm gap in musical pattern discovery

By examining the patterns extracted from the synthetic data in this section, we gain new insight into how we expect the algorithms to perform given controlled input data. For example, suppose we apply pattern discovery algorithms to a musical piece with sections of ostinatos (P_1 being a special case) and ornamental bridge (P_2 being a special case), the algorithms might not be able to return the two sections as patterns. This might come as a surprise to human annotators who are looking for obvious repetitions in the piece. By reducing the complexity of current patterns discovery corpora to simple concatenations of preselected patterns, synthetic data may be useful for better inspecting, comparing, validating, and evaluating the algorithms by modelling the most likely behaviours of human annotators. For example,

one can identify which algorithm is better suited than others for retrieving "conventional" (exact repetition) or "surprising" (regularities in randomness) kinds of patterns.

4.2.6 The ground truth and single score problem

In previous chapters, we have already seen the problem with [ground truth](#) human annotations and the nature of pattern. In this chapter, we have seen these issues in the context of algorithms again. For algorithms, data is the ground truth that imbues prior knowledge. We used different types of data—random excerpts, simple patterns, human annotations—to approximate what are and what are not patterns. This leads to a chicken-and-egg problem: we would like algorithms to perform pattern discovery automatically, but to develop and assess a pattern discovery algorithm or system, one has to have a certain amount of data already in which we know what the patterns are and where they are. There is also tension between the generality of patterns and algorithms and the specificity and discriminative power required for the tasks.

In MIR, other tasks such as musical similarity, automatic chord estimation, and music segmentation face similar problems. In fact, this problem is also sometimes seen in many other domains where computational methods are employed, such as recommendation algorithm design, mood and emotion detection, medical and judicial tracking and tracing. What should be taken as the "ground truth" data for establishing an effective computational system? Does a "universal" ground truth exist and, if so, is it feasible to obtain?

The full debate on the correctness of using a "ground truth" is an important one, but we will focus on the case of musical pattern discovery. Consider the following scenario: if there is a specific musical pattern entry in the ground truth dataset, it may not be entirely fair or desirable to require a pattern discovery system to spot this pattern without regard for what precedes and succeeds this pattern, without regard for who is using the system, and without regard for how we could leverage our domain knowledge to inform the users why the patterns are retrieved. Therefore, the question of the ground truth could be better reasoned and answered with considerations such as, what is our input data? Under what circumstances will the system be used? What are the desired behaviours? What is the expected output?

A related issue is that of a single score evaluation or objective. Using a single score dictates that higher equals better, which is rarely the case. One of the underlying conceptual problems is the assumption that finite sized datasets are all there is. Dy-

namic aspects and sub-populations of the dataset are often not incorporated into the considerations.

Some common ways to mitigate these issues and the brittleness of the algorithms include out-of-distribution testing (e.g. using different types of music from different distributions), domain transfer (e.g. using synthetic data), reduce training data (e.g. using a small portion of all the training data available), and multiple-task testing (e.g. using many tasks to evaluate). We have already touched on several of these in this dissertation so far. In addition, we briefly discussed the possibility of personalisation of pattern discovery algorithms in previous chapters. Here again, we emphasise that the output of different algorithms might be useful for different application scenarios. To give a hypothetical example, one algorithm could be more suitable for extracting patterns for educational purposes, and another could be more suitable for improving the segmentation task in MIR, with the two algorithms having totally different outputs. There may be overlaps between different application scenarios where the output patterns can be used for multiple purposes. However, there is a lack of an agreed-upon approach to categorising and formalising different pattern extraction algorithms according to their potential applications, which should be a future direction for advancing the field.

4.3 Summary

A massive outpouring of intellectual and practical interest in pattern discovery algorithms has been precipitated by the development of machine learning methods, especially connectionists' deep learning methods. This introduced a range of generic pattern discovery algorithms in addition to the domain specific musical pattern discovery algorithms. We propose to evaluate musical pattern discovery algorithms computationally in five ways: visualisation, [PP](#), [CC](#), and synthetic data with planted patterns. These new methods serve to better enable the inspection, comparison, validation, and evaluation of these algorithms beyond the limitations imposed by using numerical evaluation metrics such as accuracy and [F₁ score](#). In addition to extending previous evaluation methods, we also provided examples and analysis on the prominent challenges and practical bottlenecks in the field.

More specifically, we first proposed ways to visualise the positions and features of musical patterns. Following which, by leveraging data fusion and machine learning methods, we found discrepancies between the output of different pattern discovery algorithms, as well as between algorithmic output and human annotations. Rhythmic aspects have the most influence on how well the classifiers perform at distin-

guishing between different groups of patterns. Based on these findings, we then proposed the use of synthetic data, evaluating the pattern discovery algorithms in a simpler and more controlled manner. We arrived at the discussion about ground truth, and identified the tensions between the generality of the concept of pattern and the specific tasks. Our evaluation results vary on the limited datasets available, but each of them provides a new means for investigating musical pattern discovery algorithms, namely, by visualising them, combining them, differentiating between their output, and synthesising artificial input.

4.4 Discussion

This section will discuss and summarise a few topics we touched on about musical pattern discovery algorithms. They were not crucial discussions in the main sections of this chapter, but are still important topics.

Different two types of pattern discovery

From the establishment and occurrence metrics we introduced in Section 4.2.1, in combination with the discussions about patterns in Chapter 2, we can see a division of pattern discovery into two subtasks: one is a strict query or search task to find occurrences of the units, and the other is a flexible exploratory task to establish the units. We have already mentioned how they could be linked, now we will also discuss how they are different for some discussions in future chapters.

In the strict case, we may strive to eliminate subjectivity and ambiguity in the computation steps. We assume that one has the prior knowledge of a descriptive, deterministic, precise definition of what consists of a pattern and its occurrences. Given what these prototype units are and how the occurrences of the pattern can vary, a computational approach could help with automation and speed up the process: once the units and desired variational relations are encoded, a system can be designed to implement such sufficient conditions, and retrieve the required patterns. For example, such a system may find all exactly repeated unit of the "CDE" pitch sequences in a music piece.

In the flexible case, uncertain elements such as ambiguity and subjectivity come into play. One starts with the discriminative, probabilistic, or a fuzzy idea of what consists of a pattern. By constraining on the necessary or even looser conditions, one designs and expects the algorithms to extract unknown information from data, and compute the emergent patterns to give inspiration and new insights about the

data. For instance, using supervised and unsupervised machine learning methods, such a system may produce several pattern candidates that resemble each other.

The term "pattern discovery" is sometimes used exclusively to refer to the flexible case. However, in this dissertation, we include both types as pattern discovery—the task of identifying the units and the compositions of the units in a structure. We include both types for two reasons. The first reason is that they are linked to one another: the occurrences establish the patterns, and the patterns are repeated and varied and become different occurrences. The second is that, with both types, we also include pattern recognition along with pattern discovery, as mentioned in Chapter 2. In addition to the definition quoted in Chapter 2, let us examine a definition from the *Encyclopedia Britannica*, in which pattern recognition is defined as below:

In computer science, the imposition of identity on input data, such as speech, images, or a stream of text, by the recognition and delineation of patterns it contains and their relationships (Britannica *et al.*, 2013).

We generally agree with this definition, which does not address the difference between the establishment and occurrence. We do think that the example of music can be added in this definition.

If there are systems and algorithms addressing the two different tasks separately, they should be accessed differently. For the former, one would like to check for the correctness of the output to see that there are no bugs in the system that give unwanted output. For the latter, the coverage of musically meaningful and inspiring patterns is more valuable. What is more commonly seen is a mixture of the two tasks. Many algorithms and systems are set out to strike a balance between the two tasks to find both correct and novel patterns, with different degrees of exactness and flexibility.

A balance could be attained by involving human-machine interactions where there is a feedback loop from the human annotation behaviour to algorithmic output and from the algorithmic output to human behaviour. In this way, it is up to the user to decide when to be strict and when to be flexible and to what degree. Similarly, when interactions between humans and algorithms are possible, the different output patterns caused by ambiguity could be appreciated, and subjectivity could become a valuable first step to personalised pattern discovery.

Limitations, future work and outlook

In future work, this research could be extended in several directions. Firstly, the coverage of the algorithms could be extended. We did not perform a comprehensive

comparison of all musical pattern discovery algorithms given limited availability and format compatibility. Secondly, for synthetic data, we can add more structures, possibly nested, during the data synthesis process. Adding rhythmic variations and polyphonic patterns would make the process suitable to investigate the algorithms that consider these aspects. Lastly, a natural next step is to integrate our findings into pattern discovery algorithm design. This could be difficult, however, because the concrete steps needed to improve an algorithm are contingent on its inner workings, which may differ from algorithm to algorithm. In addition, there is often not a single way to improve a given algorithm, but rather multiple strategies that can be used to address the issues we have identified. For example, an algorithm designer might add extra filters based on our classification results, consider rhythmic aspects to a greater extent, or provide pre-configured parameter sets optimised for different application scenarios.

On a more general note, a mapping between different algorithms and the application scenarios they are best suited to would be desirable, considering the diversity of pattern discovery tasks. The disagreement we found between algorithms gives support to this claim, with the differences serving as a guide for selecting algorithms based on different applications. For example, the algorithms that discover long and overlapping patterns might be more suitable for compression, and the algorithms that discover shorter patterns might be more suitable for music-theoretic and educational purposes.

Switching from the perspective of the user to the designer, pattern discovery algorithm designers would also benefit from having the application scenarios as clear goals; for example, subsequent classification, segmentation, prediction, generation, and compression tasks. Alternatively, as mentioned above, one user-friendly design would be to have different preset parameter settings for each of these different application scenarios.

Starting from a concrete application scenario would help to clarify which classes of patterns one wishes to extract, which would, in turn, give better grounding to different evaluation methodologies. For example, the patterns extracted for a compression task might be very dissimilar to patterns extracted for generation purposes. Although the tasks we have listed so far are possibly not complete, and we do not even have the means to make a complete list of all the tasks possible for musical patterns. Future advances in open-ended sub-/task and sub-/goal discovery could provide more directions in this respect. The evaluation of pattern discovery algorithms can, therefore, be diverse and go beyond a few numerical metrics and the manual annotations of a single human expert as the ground truth.

One crucial link for diversifying the evaluation step is to bridge the gap between the application scenario and how we formulate the strategies used in our data synthesis approach. With careful design, the data synthesis approach can potentially be tuned to simulate different application contexts. Application-driven thinking combined with the data synthesis approach also has the potential to save time when it comes to annotating the data and address issues pertaining to ambiguity in the annotated patterns. To generate these high quality synthetic data, future work may take fidelity, diversity, and generalisation performance into account, as proposed in (Alaa *et al.*, 2021).

Musical pattern discovery is an interdisciplinary area of research. Over the years, we have seen exuberant interests and considerable advancement of the **SotA** from contributors with diverse backgrounds. This has brought unique challenges as well as opportunities with a diverse range of tools, formulations, and evaluation methods, including but not limited to the ones proposed in this paper. We expect that greater coherence will be created by the feedback loop between such multifaceted evaluation results and the algorithms performing on diverse types of data. More concretely, users with access to a greater range of algorithm evaluation methodologies will be better equipped to provide feedback to algorithm designers that will ultimately be used for the design of those algorithms.

Chapter 5 Computation, Functional Programming, and Music

It is indeed a surprising and fortunate fact that nature can be expressed by relatively low-order mathematical functions.

– Rudolf Carnap

In conceptual art the idea or concept is the most important aspect of the work . . . all planning and decisions are made beforehand and the execution is a perfunctory affair. The idea becomes the machine that makes the art.

– Sol LeWitt

Having familiarised ourselves with musical pattern discovery as a research area, we can now ask ourselves the question: can we do better by incorporating new perspectives into the implementation code that underlies the algorithms, in terms of the language we use to program the computation, and if so, how? In this chapter, we examine some new directions—research at the intersection of [Functional Programming \(FP\)](#) and music. We start with a broad overview of the topics we touch on in this chapter and the chapter’s structure. We will visit many connections already been made between the functional programming language Haskell and music. Informed by the advantages of Haskell, we exhibit a range of available theories and tools for linking musical patterns with transformations: laying the groundwork for our implementation and experiment in the following chapters.

5.1 Overview

We have seen programs interacting with the music domain since the advent of computing, when Ada Lovelace wrote "the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." (Lovelace, [1843](#)). Meir Lehman once said that "Any program is a model of a model within a theory of a

model of an abstraction of some portion of the world or of some universe of discourse." (Lehman, 1980). No wonder that the abstract side of music turns out to fit so well with the logic of all kinds of programs.

All the programs and analysis we have done so far have this added complexity of the program implementations in addition to the algorithms. In the same work, Lehman also pointed out 5 laws of program evolution: the law of continuing change, of increasing complexity, the conservation of organisational stability, of familiarity, of program evolution. Without going into detail about them, we list them here to point out that the concrete implementations can be complex and very relevant to a research idea. By looking more closely at the implementation side of things, we would like to explore more fitting models for musical patterns and the implementation of their comparisons.

Music DSLs and software

Let us take a broader look at the specific programming languages and software in the music domain. There are many existing software and DSLs in music and a constant stream of new ideas have been implemented.

For example, in the 1980s, Music Mouse—an algorithmic musical composition software and the Hierarchical Music Specification Language (HMSL) as a music programming language were created. The Music Macro Language was a music description language used in sequencing music on computer and video game systems. These languages were created by taking ideas from music and trying to communicate with machines to express these ideas. For instance, HMSL's main interface was designed for the manipulation of musical parameters using "the metaphor of shapes, which can be created, altered, and combined to create a musical texture".

A range of successful programming languages have been used in creating and performing music. These names might sound familiar to both musicians and programmers: SuperCollider, "a framework for acoustic research, algorithmic music, interactive programming and live coding"; ChucK "A strongly-timed, concurrent, and on-the-fly audio programming language", Pure Data "a visual programming language developed for creating interactive computer music", and similarly, Max/MSP.

In the context of music analysis, there are languages and applications such as Rubato and CHARM (Common Hierarchical Abstract Representation for Music). Developed in (Pearce, 2005), CHARM is "a logical specification of an abstract representation of music for use in a wide range of areas including composition, analysis and archiving". In (Janin, 2016), the following comments about music programming languages are made:

Every music programming language induces a music algebra that defines both the basic objects and the combinators that allow to build complex music objects from simple ones.

In the same paper, a "high level [DSL](#) that is both hierarchical and modular" was implemented and used to model music in the functional programming language Haskell. Based on the discussion we had in Chapter 2 about patterns, their combinations, and related concepts, we can hopefully start to see the relevance of programming languages to musical patterns.

Functional programming

Functional programming, as a programming paradigm that puts focus on modularity, compositionality, declarativeness, and type safety, has seen much success with music. In an array of functional programming languages, Haskell is a statically-typed, lazy, purely functional language that we have chosen to explore. Haskell is widely used in [Programming Language \(PL\)](#) research, and it enjoys a reputation for being powerful and flexible when it comes to abstracting and implementing useful programming structures, as well as the steep learning curve and esotericism that comes with it. With the [Glasgow Haskell Compiler \(GHC\)](#), there are many packages and [FP](#) techniques that have become available. From [MIR](#) more specifically, work such as (De Haas, 2012) inspired us to see if this powerful language can assist in the issues we face in musical pattern discovery.

Some benefits that come with good implementations

What benefits could we potentially gain by discussing more about implementations for our computational and musical patterns? It has been observed that "Haskell is often called an executable specification language; i.e. programs serve the role of mathematical specifications that are directly executable". In (Petricek, 2019), it was argued that "If you structure your programs well, a mathematical certainty could be obtained by reasoning about the programs", which has appeared in a range of work, as early as (Floyd, 1967). The pursuit of mathematical certainty in programs has been divided largely into three directions: logic, type theory, and category theory. Although we do not make deep connections with these theories in this dissertation, we will give a broad view here regarding one of the above—type.

Type systems are designed for classifying values. In his seminal book (Brady, 2017), Edwin Brady defines type-driven development as "an approach to coding that embraces types as the foundation of your code, essentially as built-in documentation

your compiler can use to check data relationships and other assumptions.". (Zalta *et al.*, 2005) mentioned that "To a large extent, types determine the level of abstraction of the language". By using types, we can potentially construe implementations in a more principled way.

Types can prompt us with meaningful constructions in our domains and prevent meaningless ones. It has been said to be the most popular lightweight formal method. Some type systems serve more as a documentation tool, some support better tooling, and some use types for complex interactions between parts of programs. In fact, type theory is an vibrant area of research where recent work on refinement types and dependent types allows for more detailed specifications of types; optional and gradual typing allows for different degrees of mixture between static and dynamic checking.

There is a saying that "well-typed programs cannot go wrong". Both the type checking and construction mechanisms of type systems lead to better implementations of computational ideas. However, there is only so much a type system can help with. A quote from Turing about computational errors or when a program does "goes wrong" could be fitting here: "errors of conclusion can only arise when some meaning is attached to the output signals from the machines" (cited in Turner and Eden, 2008), which points to the fact that software can only malfunction but cannot ever dysfunction. This is in accordance with what we have seen in previous chapters that, if we cannot anticipate all the specifications we want from new data, we will encounter unexpected output from algorithms. Finding a precise computational solution for musical pattern requires synergy of many areas.

Musical patterns recontextualised

From previous chapters, we have looked at musical patterns from both the empirical and formal point of view, which have been of interest to musicologists, music theorists, cognitive scientists, and psychologists. In the context of computing, within the research area of MIR, many algorithms have been designed to retrieve musical patterns. Because "musical pattern" as a terminology has been used in various contexts, it is not entirely clear if it is possible to give a precise answer to the question: what constitutes a musical pattern? By making use of PL features, we strive to find new ways to describe the musical patterns and their repetition and variation in a computational setting, where more complex procedures can be created and analysed.

This chapter

In the rest of this chapter, we will examine an array of applications of Haskell¹ We will see synergistic cases between programming and music generating new insights and applications. By connecting with FP, we hope to bring in a new perspective by realising that we can use transformations to compare musical patterns.

5.2 Functional programming language Haskell and music

There are over a hundred music-related packages available for Haskell (“packages by category | Hackage”, 2020), including the performance-oriented live coding package Tidal (McLean & Wiggins, 2010), the general-purpose library Euterpea with recent improved composition and improvisation capabilities (Quick, 2018; Quick & Thomas, 2019), and the theory-based libraries such as Mezzo and HarmTrace (Magalhães & Koops, 2014; Szamozvancev & Gale, 2017).

Although each package uses different types and functions to represent the elements and operations in music, all of them make heavy use of the notions of pattern, structure, and rules in music. Musical parallels of certain programming techniques used in these packages can also be potentially used to teach abstract programming ideas. In fact, music-related packages are being used as pedagogical tools such as in (Aaron, 2016; Hudak, 2000). In addition to the conceptual parallels, new art forms can be created (McLean, 2014; Quick & Thomas, 2019) and more analysis of art can be carried out (Magalhães & Koops, 2014; Melkonian *et al.*, 2019).

In the remainder of this section, we will examine several Haskell music packages to see how they model music and patterns. In the next section, Section 5.3, we will use what we learned to propose approaching musical pattern discovery using transformations, that can be naturally modelled by functional programming.

The Haskell School of Music

Initiated by Paul Hudak, Euterpea is a widely known package in the functional art community. It provides utilities in algorithmic composition and analysis, MIDI handling, audio processing, sound synthesis, and instrument design. Accompanied by The Haskell School of Music textbook, it also has a companion library, HSoM,

¹We provide an introduction to Haskell and category theory in Appendix B and Appendix C for readers who do not have a high degree of familiarity with the language. We hope that it could be helpful to refer to during this chapter and the next.

which includes a musical user interface that consists of "a set of computer-music specific GUI widgets for designing real-time, interactive musical applications". Pattern-based music generation have been explored in (Quick, 2018) with this library.

Euterpea uses recursive types to represent music. In a different but related package, haskore (Hudak *et al.*, 1996), the operations to transform musical objects are also defined in the Music type.

The idea is to make it customisable so that more customised modifiers can be added. The idea of phrase, which is related to musical patterns, is also encoded in this type.

Tidal

Tidal was created to facilitate the "performances with patterns of time" (McLean, 2014). The package, also known as TidalCycles, provides "a live coding environment designed for musical improvisation and composition". The author made pattern-related observations while making this package and stated that "pattern pervades the arts". The idea is that a pattern library should contain a range of basic pattern generators and transformations, which can be straightforwardly composed into complex structures, and it may also contain more complex transformations, or have a community repository where such patterns may be shared.

Live coding, which can also be referred to as on-the-fly programming, is often used to produce sound and images in real-time by writing code on stage. Using Haskell for this purpose has a few benefits, including the language's terseness and focus on compositionality. It guarantees safe updates and provides strongly typed user-side libraries.

HarmTrace

HarmTrace is a library that models functional harmony and has been used for analysis and generation purposes (De Haas, 2012; Koops *et al.*, 2013; Magalhães & Koops, 2014). The model encodes a musical piece recursively with Tonal(Ton) and Dominant(Dom) phrases. Using GADTs, the package models different transitions in harmonic categories for parsing and creating the harmony automatically. We do not go into detail with this package because we do not look into polyphonic pieces.

Haskell Music Theory (HMT)

The HMT library includes implementations for many ideas in various music theories, such as (Lewin, 1979; Meyer, 1929), and with modules named after the theorists

such as Lewin and Forte. For example, the package calculates the relatedness measure as defined in (Lewin, 1979) It uses functional concepts such as fold, map, and list comprehension to compute the mathematical concepts in such theories.

Mezzo

Mezzo is "a Haskell music composition library that enforces common musical rules in the type system". A slogan proposed was "well-typed music does not sound wrong". The package uses lots of recent features from the type system of Haskell to check counterpoint rules. Almost all music vocabulary are defined in types. Constructs such as type class, GADT, and type family are widely used in this library. To relate the types, 87 type families are used in categorising intervals and chords. An additional 31 type families are used for primitives to define vector-like operations. The rules of counterpoint are encoded in type classes and the error messages are also customised.

This Mezzo package is no doubt an impressive usage of the powerful type system. However, as many questions in music do not have clear right or wrong answers, there has been limited application of this package. An overuse of a feature in a programming language can also sometimes come with the price of being cumbersome. Nevertheless, in situations where analytical rules are important and hard to check manually, this type of DSL can undoubtedly be useful.

Other packages and what we can learn from them

Other Haskell packages also provide interesting perspectives for computation in music, such as probabilistic temporal graph grammars (Melkonian, 2019) and the functional reactive programming framework (Nilsson & Chupin, 2017). It will be practically impossible to review all these packages.

From what has been examined above, we can already see the power of using domain-specific knowledge to write and guide a variety of calculations. A shared language for both humans and computers to understand and communicate is key.

For our purpose, we could not find a package dedicated to the comparison and examinations of musical patterns, especially between the algorithmically extracted and the human-annotated ones. In the next section and the following chapters, we will propose using a range of musical transformations for this purpose, and write our own DSL—Pattrans. Before this step, we will use the remainder of this chapter to examine literature regarding transformations.

5.3 Approach musical pattern discovery using functional programming: transformations

Having examined some connections between functional programming and music, in this section, we propose to make connections between patterns and transformations, which will put us at an advantage while exploring patterns using the functional programming language Haskell. The rest of this section invokes literature from music and pattern recognition to discuss the relationship between patterns and transformations.

5.3.1 Our proposal: Pattern and transformation

Just as "pattern", "transformation" is a common term used in many contexts to signify the process behind a change or difference. To refer to the [OED](#), taking the most relevant three definitions of transformation: its general meaning is a "A marked change in form, nature, or appearance.;" in mathematics logic, it is "A process by which one figure, expression, or function is converted into another one of similar value.;" in linguistics, "A process by which an element in the underlying logical deep structure of a sentence is converted to an element in the surface structure.".

In our context, transformation refers to the process by which one pattern transforms into another. Unlike pattern, transformation abstracts away the content of patterns and focuses on the *relations* between patterns, which can be crucial to musical pattern discovery. This realisation is inspired by functional programming, where functions, or transformations in a sense, are treated as first-class citizens. Furthermore, as far as we know, we do not see a systematic investigation linking musical pattern discovery and musical transformations. We, therefore, emphasise this connection in this dissertation.

As a piece of side information, we can regard this relation as an equivalence relation computationally, but not strictly musically. Computationally, we have a set of pattern occurrences to compare, and each transformation can relate a prototype pattern to other occurrences, and this creates an equivalence class for each transformation. In music, however, it is not necessarily true that all relations between pattern occurrences are transitive, as we discussed in Chapter 2.

Some examples of transformations are shown in Figure 5.1, where we see a pattern being transformed in three different ways. Different transformations can thus be used to model different repetitions and variations.

5.3 Approach musical pattern discovery using functional programming: transformations

Using transformations to model the relations between different patterns is not a new idea. In fact, generally, relations between the properties of entities have been shown to be an important aspect to consider (Milewski & Tabachnik, 2018). Our focus here is to use transformation to help us formulate musical patterns further and use this formulation to compare patterns annotated by different agents, as we discussed in previous chapters.

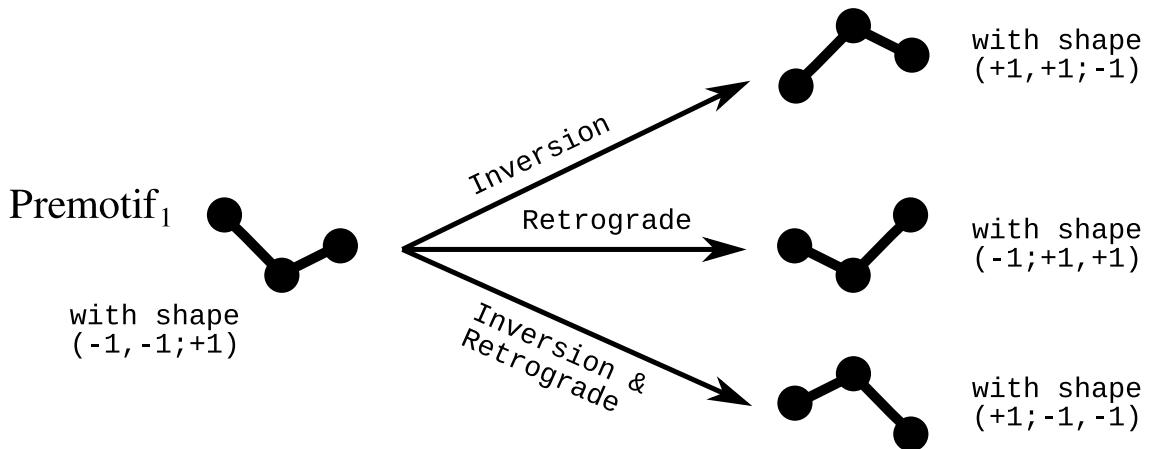


Figure 5.1: Examples showing three transformations applying on a pattern. Reproduced from (Buteau & Mazzola, 2000).

Furthermore, we would like to leverage functional programming and encode the transformations and their combinations in Haskell. By encoding transformations in a functional style, one gains the advantages of:

- Predictable properties when composing transformations
- Potentially fewer, more descriptive lines of code
- More robust implementation thanks to type checking

To gain a broad view of musical transformations and transformations in general, we will examine a range of literature on the topic of transformations. We will again consider the human perception aspects, the relation between transformation and similarity, and arrive at the strengths and limitations of our proposal to use transformations for investigating patterns.

5.3.2 Musical transformations

In this section, we examine and give examples of musical transformations. Different transformations have been used in different types of music. Across a wide range of corpora in various styles, transformations have also been used to analyse music in the context of music theory and musicology. Described both colloquially and mathematically, there exist both generic transformations (such as repetition and trans-

position) (Ockelford, 2017) and specialised transformations (such as a $[+1, +3, +2]$ semitone transposition for three-note sequences) (Lewin, 2008). These have been analysed manually by musicologists and music theorists before the advent of computing devices.

There are many examples of transformations in Western music, such as those by Mozart, Bach, Haydn, Beethoven, Berlioz, Wagner, and Brahms. In Figure 5.2, we show an example of retrograde from Handel: the second half of the music can be viewed as a transformation of the first half. Retrogrades, or palindromes, can also be seen in multiple compositions following twelve-tone techniques and mirror canons. As another example, in Figure 5.3, we can see short pitch patterns in Gregorian chant notations, and the relation between the first and second column can be viewed as a type of transformation. The tense and detense relations can be viewed as inversion, or in some cases, also as retrograde.

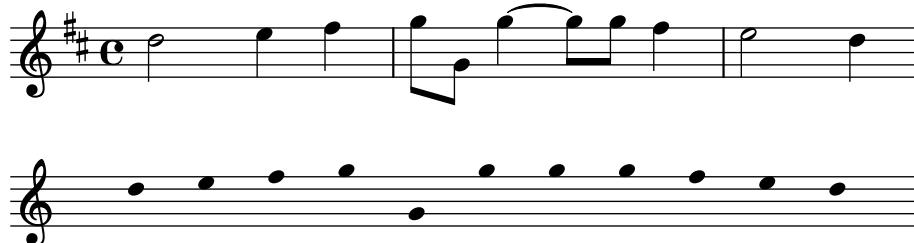


Figure 5.2: Retrograde in Handel's Messiah, Hallelujah chorus. Reproduced from (Ockelford, 2018).

We must contend with the varied terminology used to describe transformations, just as with the concept of pattern in Chapter 2. For example, (Buteau & Mazzola, 2000) employs a set of musical transformations, such as inversion or retrograde, and names the transformed premotives as a gestalt. Another example is that of a thematic transformation, which is "used to define the process of modifying a theme so that in a new context it is different but yet manifestly made of the same elements; a variant term is 'thematic metamorphosis'", according to (Macdonald, 2001). As the case with pattern, for more generalisability, we do not make these distinctions. We do examine a few more types of transformations below in different types of music and different traditions to gain perspective on the role of transformations in music.

Transformations in folk songs

According to (Nettl, 1956), transposition is common and structurally important in folk songs from various cultures:

melodic sequence, or, in a more generalized sense, transposition of a section to different pitch levels... is found in almost all cultures, sometimes as

TENSE		DETENSE	
BINARY			
A	● ● ● (scandius)	a	● ● ● (climacus)
B	● ● ● (torculus)	b	● ● ● (porrectus)
TERNARY			
C	● ● ● ● ●	c	● ● ● ● ●
D	● ● ● ● ●	d	● ● ● ● ●
E	● ● ● ● ●	e	● ● ● ● ●
F	● ● ● ● ●	f	● ● ● ● ●

Figure 5.3: Short pitch patterns in Gregorian chant. Reproduced from (Monelle, 2014).

a short, incidental bit, sometimes contributing to the structural organization of the whole piece, sometimes even dominating it. It may be exact or approximate transposition; it may be accommodated to the tonal system already established or give rise to the formation of a special scale.

The paper regards the basic function of transposition as "not unification but elaboration and variety". In our case, this points to the repetition and variation we have been talking about. The author comments further on the role of transposition "the two principles (repetition and variation) involved are not only contrasting but also complementary. Transposition has two aspects: it repeats a melody and it also changes it. And this is true of variation of any sort.". This links to our discussion in Chapter 2, and it seems to us that the transformation approach can be used to investigate the repetition and variation in folk songs.

Repetition and transformation

As mentioned above, transformation can be used to model repetition and variation. In (Margulis, 2014), it was proposed that the amount of repetition varies between compositions created by different composers in different styles; in (Middleton, 1990), it was proposed that popular music is highly repetitive. In (Ockelford, 2017), the zygonic model was proposed, which focuses on imitation, and therefore to a certain

extent, repetition and variation. These musicological enquiries into repetition and variation can be supported with the transformation approach: if the above quantitative arguments are true, we should be able to find supporting evidence in the patterns and their occurrence relations using transformations.

Transformation and Inganno

Closely related to (Schubert, 1995), the word fuga in the late Renaissance embraces not only pitch interval of imitation, but also other variation techniques that are applied to melodic material (inversion, retrograde, and inganno). Fuga d'inganno is a pitch interval change that keeps the original's solmisation syllables. Due to the fact that solmisation names can designate two or three pitch classes, a wide range of melodic variations are possible. In this dissertation, we do not have a focus on this type of transformation. We do, however, acknowledge the different types of richness that transformations can bring.

Transformation in mathematical music theory

In (Mazzola, 2018), a range of transformation-related ideas were articulated in the context of mathematics:

Our taxonomy yields two types of paradigms: by transformations and by similarity—however, in practice, they often appear in mixed form. In a mathematical perspective, the first type is covered by group theory, the second by topology. This means that fuzzy concepts in the humanities are not *a priori* useless, they can be incorporated into exact reasoning by means of a refined paradigmatic reconstruction. Transformation as a transfer of association to generative rules.

We will briefly mention the mathematical perspectives on transformation and the relation between transformation and similarity later in this chapter. We also support the thinking that using transformations can turn fuzzy associations into more useful computational rules.

Lewin

(Lewin, 1987) is a seminal work that is fundamental to many mathematical approaches to music analysis which originated in the 1980s. The transformations considered in this work seem to have a parallel with functional programming: "A Lewinian transformation is nothing more or less than what mathematicians call

5.3 Approach musical pattern discovery using functional programming: transformations

a function" (Hook, 2007). It supports us more that we are using a programming paradigm with the focus on functions to model transformations.

Figure 5.4 and Figure 5.5 shows one of the transformations considered in Lewin's work, demonstrating "how Bach chains MUCH and RICH in the first Two-Part Intervention." (Lewin, 1987). RICH denotes a retrograde-inversion chain and MUCH(s) is the retrograde-inverted form of s with the beginning overlapping the ending of s to the maximum possible extent.

Although our aim is not to computationalise Lewin's theory, we can see from this example that the complexity of using transformations can be high, which can be assisted by computational methods. At the same time, we need to be careful about the stacked complexity as we considered for other algorithms in Chapter 4. We will discuss more about the limitations of this approach in Section ??.

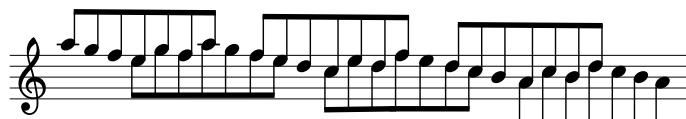


Figure 5.4: A excerpt of music that can be analysed with transformations. Reproduced from (Lewin, 1987).

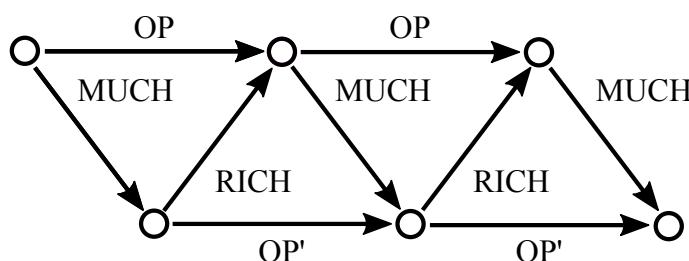


Figure 5.5: The RICH and MUCH transformations in Figure 5.4. Reproduced from (Lewin, 1987)

Twelve-tone techniques

Twelve-tone techniques in atonal or serial music use the notion of transformations. The music is often constructed by choosing a tone row (a non-repetitive ordering of a set of pitch classes) as the basis of the piece, which is called the prime series P . Given the twelve pitch classes of the chromatic scale, there are $12! = 479,001,600$ tone rows.

Appearances of P can be transformed from the original in three basic ways (Argentino & Mackenzie, 2019):

- transposition up or down, giving P_x
- reversal in time, giving the retrograde R

- reversal in pitch, giving the inversion I

For this type of music, taking a transformation-based approach is perhaps non-negotiable. In the same context, a connection has been found between musical transformation with transformations in languages: (Argentino & Mackenzie, 2019) demonstrated that "the structure of these transformations (in serial music) parallels language games commonly found across languages". Considering the transformations in languages will be beyond the scope of this chapter. However, through connections such as this, we do realise that a transformation-based approach can benefit from and be useful to other domains as well.

Transformations in an educational context

In a musical educational context, Laurie Spiegel and her 1981 paper "Manipulations of Musical Patterns" lists 12 transformations (Spiegel, 1981). Taken from Spiegel's own introspection as a composer, the 12 classes of pattern transformation are:

- transposition (translation by value), reversal (value inversion or time reversal),
- rotation (cycle time phase), phase offset (relative rotation, e.g. a canon),
- rescaling (of time or value),
- interpolation (adding midpoints and ornamentation),
- extrapolation (continuation),
- fragmentation (breaking up of an established pattern),
- substitution (against expectation),
- combination (by value-mixing/counterpoint/harmony),
- sequencing (by time-editing)
- repetition.

Spiegel felt these to be "tried and true" basic operations. We will consider many transformations in this list in the next chapter where we implement our own DSL. Furthermore, Spiegel proposed that studying these transformations could aid our understanding of the temporal forms shared by music and the human perception of them. We will touch on the aspects of human perception of transformations in Section 5.3.5. In the next section, we will examine some more literature in relation to musical pattern discovery algorithms.

5.3.3 Implicit and explicit transformation in musical pattern research and algorithms

We have seen growth in research focusing on the connections between human expertise and new machine learning algorithms, which are often black-box optimisation models. Different paradigms, such as active learning, curriculum learning, continual learning, and many other sorts, can benefit from extra knowledge imbued into the system. Transformations can potentially be used as an encoding of human priors or oracles to guide these algorithms.

Some musical pattern discovery algorithms explicitly consider transformations to be crucial and explicitly implement transformations, while some others take a "good-to-have" stance. Geometric methods, in particular, consider the notion of transformation explicitly: "... certain types of thematic modification (e.g., diminution, augmentation, inversion, reversal and transposition) correspond directly to geometrical transformations." (Meredith *et al.*, 2002). In some other algorithms introduced in Section 2.4.1, we have already seen similar considerations such as those in the suffix arrays.

The recent work of Patternfinder (Garfinkle *et al.*, 2017), written in Python, also employs the notion of transformations. Certain transformations such as real transposition, augmentation, and diminution are considered, but Patternfinder has not been used to analyse pattern discovery algorithms.

In (Conklin, 2002), the idea of transformations and invariance were also articulated:

One of the basic principles for automated pattern discovery proposed by this paper is "to detect common structure, they (patterns) should not be affected by typical musical transformations such as transposition and simple melodic elaboration".

(Lartillot, 2004) acknowledges that "Usually, patterns are not exactly repeated but feature numerous kinds of transformations." The concrete considerations in the paper, however, only appear in the Future Work section: "(with adaptation of this model)... diatonic transposition, which is a typical motivic transformation, would easily be identified". It also explicitly states that "Different kinds of transformations, such as inversion and retrograde, should be considered as well."

In (Stech, 1981), patterns were located up to their exact repetition, inversion, retrograde, and retrograde inversion. A Fortran program was used to analyse musical examples from the works of Dunstable, Hindemith, and Varese.

Algorithms in other domains, particularly in the vision domain, have also begun to pay more attention to the underlying transformations in images (Cohen & Welling, 2016). To be precise about the transformation considered in the musical pattern discovery algorithms, we focus on two families of algorithms and applications below.

Geometric algorithms

To elaborate more on the geometric family, (Meredith *et al.*, 2002) gives examples and descriptions of transformations such as inversion, retrograde, augmentation, and diminution. As the paper proposes the geometric method for pattern finding, transformations are also described in this geometric context in particular, for example:

If we represent a musical passage as a set of points in a two dimensional space in which the dimensions represent pitch and time, then the musical transformation of reversal corresponds to a geometrical transformation of reflection about an axis parallel to the pitch axis.

There are many other ways in which a musical pattern may be modified to give a second pattern that is perceived to be a version of it but even from the foregoing discussion it should be clear that the class of perceptually significant musical repetitions is a very diverse set.

We have also shown that if the music to be analysed is appropriately represented as a set of points in a multidimensional space, then certain types of thematic modification (e.g., diminution, augmentation, inversion, reversal and transposition) correspond directly to geometrical transformations within such a multidimensional representation.

This paper also points out an algorithm that does not explicitly consider a type of transformation (transposition invariance).

Hsu *et al.* (1998) used a dynamic programming technique to find repeating factors in strings representing monophonic melodies. Though having an $O(n^4)$ worst case time complexity, it works much more efficiently in practice. In their study, Hsu *et al.* did not consider transposition invariance, but this can be obtained straightforwardly by using an interval string representation instead of an event string representation.

With these considerations behind the geometric family of algorithms, we should be able to use transformations to compare the patterns discovered by these algorithms. This comparison is what we will attempt to do in Chapter 7.

In pattern viewer and searcher applications

Although the pattern viewers discussed below are not our focus for comparison in this dissertation, there are connections to be made between a pattern search problem and a pattern discovery problem, as we discussed in Section 4.4. On recent developments in browsing patterns, (Klaus Frieler & Dixon, 2018) gives many pointers to online applications and their usage of transformations:

Themefinder, which interfaces with some large databases of folk and classical music in **kern format. The search works well and is fast, though it does not offer metadata filters, regular expressions, or a multi-staged search.

Musipedia is branded as a "Wikipedia for Music" and is based on a user-generated database of melodies. Search queries are given in Lilypond format, but a piano-like user interface to enter queries with the help of the mouse is also provided. The search is based on similarity matching and, hence, always fuzzy; it is not possible to enforce only exact matches.

... pattern discovery in Indian Art Music based on automatically extracted pitch contours and a large set of specialized methods. A demo for browsing patterns in a large audio corpus and a visualization of pattern networks including audio snippets can be found on the accompanying website.

We can see that functionality such as regular expressions, exact matches, and visualisation is important in this area of research. Predefined transformations can be useful as the paper goes on to describe their implementations of transformations and regular expressions:

To execute a basic search, the user has to enter a pattern as a space or comma separated list of elements and choose a corresponding transformation, that is, a mathematical mapping of the basic melodic representation, also known as viewpoints. The underlying search algorithm is built upon the basic Python regular expression module, which is fed with virtual Unicode strings constructed from the different melodic representations (transformations) with different alphabets. Currently, ten pitch-related transformations for primary search are offered (e.g., MIDI pitch, semitone intervals, CDPCX). An additional 18 transformations, such as duration, IOI classes and various structural markers, are supplied for the optional secondary search.

The terminology in use here differs with ours. The transformations here refer to a transformation in specific features or viewpoints, as opposed to how we use trans-

formations in the sense of a pattern transforming into another with transposition, inversion, and so on. There is, however, an overlap when certain changes of features can be used to specify the transformations. We will keep in mind that by focusing on different features, we can also discuss another range of transformations.

5.3.4 Generic transformations

After exploring transformations in music, we can already make some connections between music-specific transformations and some more generic ideas about transformations in mathematics as well as their application in pattern recognition. In this section, we will look at these connections to further examine the suitability of our proposal—using transformation for investigating patterns. They will not be our focus for implementation in later chapters, but they do provide some future directions and are included here for related discussions.

Symmetry

The invariant aspects of transformations take us into the realm of symmetry: there is symmetry when "under certain manipulations, namely transformations, specific features of a physical system remain unchanged" (Glattfelder, 2019). This connection was established in physics as early as in (Noether, 1918). In other words, when a pattern is unchanged under a transformation, it has symmetry. "Symmetry is a vast subject, significant art and nature" wrote mathematician Hermann Weyl (Weyl, 2015). The role of symmetry is examined in (Kempf, 1996) in the context of various types of music. It was argued that "symmetry is one of the 'universal principles (in music)' and a specific aspect of repetition". It has also become an important aspect in machine learning and, more broadly, AI research (Chollet, 2019).

One useful piece of information for our purpose comes from the fact that there are only four mathematically well-defined symmetry types in two dimensional Euclidean space (Yi, 2013) (excluding identity transformation):

- $x = x + \Delta x, y = y + \Delta y$: Translation
- $x = -x, y = y + \Delta y$: Glide reflection
- $x = -x, y = y$: Reflection
- $x = \sqrt{x^2 + y^2} \cos \frac{2\pi}{n}, y = \sqrt{x^2 + y^2} \sin \frac{2\pi}{n}$: Rotation

In music, the first three cases are covered by transposition, inversion, and retrograde. Retrograde-inversion may be viewed as a type of rotation. However, it is difficult to find a direct analogue of continuous rotation in the music. For example, the scaling from CDE to CEG can also be viewed as a type of rotation.

Another notion related to invariance is that of *equivariance*. Intuitively, it can be viewed as the relation one sees in an object before and after a camera filter: the movement of the object induces a movement of the filtered picture—the two are not entirely the same, but changes in one correspond to predictable changes in another. We can imagine that this would be something desirable for an algorithm to learn in order to discover pattern occurrences across similar pieces.

Lots of common patterns extracted from various algorithms actually satisfy certain symmetries. This has a connection with work on automatic symmetry detection such as (Yi, 2013). For us, this also points us in new directions, such as learning the transformations from data.

Group theory

Symmetry and transformations both have connections to group theory in mathematics. Group theory might help us identify additional properties of the transformations. In fact, many works investigate this topic between music and group theory.

To begin, retrograde (R) and inversion (I) have been known to form Klein four-group (Mazzola *et al.*, 2016). (Rings, 2011) applies transformational techniques to the study of tonal music and lists the group structures in transpositions and inversions:

The diatonic transpositions form a cyclic group of order 7, or Z_7 . The diatonic transpositions and diatonic inversions form a dihedral group of order 14, or D_{14} . The chromatic transpositions form a cyclic group of order 84, or Z_{84} . The chromatic transpositions and chromatic inversions form a dihedral group of order 168, or D_{168} .

In (Hart *et al.*, 2009), mathematical structures such as the Frieze groups and the wallpaper groups are used to generate musical patterns. In the case of polyphonic music, even more mathematical structures can be explored with transformations, such as in (Tymoczko, 2009).

We list these works here to show that transformations can inherit nice mathematical properties, which can be potentially used for optimising and verifying programs. This aspect is not included in our implementation for this dissertation and we reserve this for future work.

Category theory

Category theory is another mathematical area that is related to music (Acotto & Andreatta, 2012), transformations (Popoff, 2012), and functional programming (Milewski & Tabachnik, 2018). As applied category theory is a highly active research area, we will not attempt to provide a review here. We will, however, use category theory as part of our implementation in the next chapter, but not in direct connection to music and transformations.

Pattern recognition

In pattern recognition, the idea of using transformations to describe and formulate pattern recognition problems goes back a long way. The importance of *invariance* in transformations was emphasised with regard to feature extraction and distance measure in (Simard *et al.*, 1998):

use a so-called feature extractor whose purpose is to compute a representation of the patterns that is minimally affected by transformations of the patterns that do not modify their category.

... an invariant distance measure constructed in such a way that the distance between a prototype and a pattern will not be affected by irrelevant transformations of the pattern or of the prototype.

In the context of hand-written digit recognition, it was mentioned that "The key idea is to construct a distance measure which is invariant with respect to some chosen transformations such as translation, rotation and others." (Simard *et al.*, 1993). More recently, transformations have also been used to augment data for pattern recognition tasks (Ratner, Ehrenberg, *et al.*, 2017). For our purpose, although we do not augment data for training algorithms, we can use it closely with our synthetic data idea presented in the previous chapter, as shown in Figure 5.6.

5.3.5 Perception

As we did for patterns, we examine the perceptibility of transformations before we use them for finding and comparing potentially perceptible patterns. In (Hahn *et al.*, 2003), the importance of the perceptual aspects of transformations has been emphasised by several prominent scientists:

Klein, and later mathematicians (e.g. Weyl, 1952) believed that transformations have deep connections with perception, particularly in relation to aesthetic judgement. Furthermore, transformations have been viewed

5.3 Approach musical pattern discovery using functional programming: transformations

The image shows a musical score in G major with a treble clef, 4/4 time, and a key signature of one sharp. The score consists of four staves of music. Annotations are made using brackets and braces to identify specific patterns:

- Pattern1:** Occurrence 1 is indicated by a bracket under the first four measures of the first staff.
- Pattern1:** Occurrence 2 (with transposition -1) is indicated by a bracket under the measures from 9 to 12 of the first staff.
- Pattern1:** Occurrence 2 (with transposition +2) is indicated by a bracket under the measures from 17 to 25 of the first staff.
- Pattern2:** Occurrence 1 is indicated by a brace under the measures from 9 to 12 of the first staff, which also covers the first occurrence of Pattern1.
- Pattern2:** Occurrence 2 is indicated by a brace under the measures from 17 to 25 of the first staff, which also covers the second occurrence of Pattern1.

Figure 5.6: Different patterns in a synthetic piece using transpositions.

as a crucial tool in building models of the kinds of patterns occurring in the natural world – for example, in the development of Pattern Theory (e.g. Grenander, 1996) and its application to computer image processing and visual perception (e.g. Mumford, 1996).

In (McAdams & Matzkin, 2001), many perceptual aspects of how people listen to music with transformations in mind were considered. Psychological properties of musical materials, including perceptual invariance under transformation, was emphasised.

In (Fiske, 1997), an experiment was carried out to compare patterns with various transformations. The paper shows that most simple transformations preserve the perception of repetition, while some more complicated manipulations make a pattern stop sounding like a repetition. Rather interestingly, not all exact repetitions were perceived as exactly the same, and some completely unrelated patterns were perceived as related, which is in accordance with our discussions about perceptions in previous chapters.

This experiment does not address retrograde nor inversion, which are arguably the most troublesome of transformations in terms of perception. One can imagine that

the difficulty in perceiving these transformations lies in how long the initial pattern is. Different results have been reported, but generally with the positive answer that people do perceive those transformations. For example, in one experiment, (Dowling, 1972) reported that "... clearly demonstrated that inversions, retrogrades, and retrograde inversions of brief melodies can be recognized with better than chance accuracy." However, most of us have the experience that, in some cases, it is difficult to notice Bach's genius use of retrograde before it is pointed out to us.

5.3.6 Transformation and Similarity

We discussed similarity in the context of pattern in Section 2.3.1. To quickly summarise the context of transformations, we recall that there are many similarity measures in MIR, and these similarity measures are used to compute common attributes of excerpts of music numerically. The excerpts can sometimes subsequently be clustered using these similarity measures.

Using transformation, however, we would then investigate pattern occurrences in a rather binary fashion (this transformation or that transformation), which provides us with a clear view of how the occurrences can be matched via semantically meaningful, predetermined predicates. We can also extend transformation with similarity measures, which is a topic we shall explore in more detail in Chapter 7. Indeed, according to (Hahn *et al.*, 2003) the similarity between two entities is a function of the "complexity" required to "distort" or "transform" one's representation into the representation of the other: "The simpler the transformation distorting one representation to the other, the more similar they are assumed to be". They present three experiments that indicate that similarity is strongly influenced by transformation distance. Similarity and transformation seem to form yet another chicken-and-egg problem, where similarity can be measured by transformations and transformations can be defined by similarity. We will not take a strict side and agree with both arguments: depending on what kind of data we have, we can either calculate similarities for transformations or create new transformations using similarities. The latter will be explored more in this dissertation.

More specifically, in addition to the transposition, inversion, and retrograde we have seen predominately so far, we will consider the transformations that can be viewed as a type of approximation. The addition of approximation is motivated by many cases where musical ornamentation, insertion, and deletion of notes are seen in musical patterns. A list of the transformations we consider will be provided in the Summary section of this chapter and in the next chapter.

5.4 Summary

In this chapter, we started by examining a range of music software and DSLs. We introduced the potential benefits of using functional programming and Haskell, and examined several existing music-related packages in Haskell. We proposed making connections between patterns and transformations, and considered various literature regarding transformations.

With this connection between pattern and transformation, we can select certain musical transformations and devise our own package in Haskell for analysing patterns. As shown in Figure 5.7, we hope to enable a comparison between patterns and their occurrences using transformations.

Combining what we have learned from the literature, we will be using two types of transformation: one type that transforms all musical events, such as transposition, retrograde, and inversion, and another type that transforms locally, such as inserting, splitting, and deleting notes. We name the first type global transformation and the second type local. Local transformations can also be viewed as approximations, with one occurrence approximately repeating the other. We will provide a full list of all the transformations used in our Pattrans package as well as their description and implementation in the next chapter.

5.5 Discussion

In this chapter, we have seen a mixture of ideas from software technology, MIR, music theory, and many other areas of research. The deep connections between the two forms of abstraction we have discussed so far—the "nouns" (abstract entities, patterns) and the "verbs" (abstract actions, transformations)—can be seen in many domains. There are many exciting ideas we can take from one domain to another: initially devised for reliable software and abstracting common structures in general, the concepts of types and functions can be potentially used to advance musical pattern discovery research. In this discussion section, we will discuss a few more issues about transformations, programming language choice, and the possibilities of cross-pollination between disciplines.

FP and ML

Functional programming features and software engineering considerations are gaining a stronger and stronger presence in machine learning. More emphasis has been placed on the robustness and verification of AI (Research, 2019). In (Balduzzi

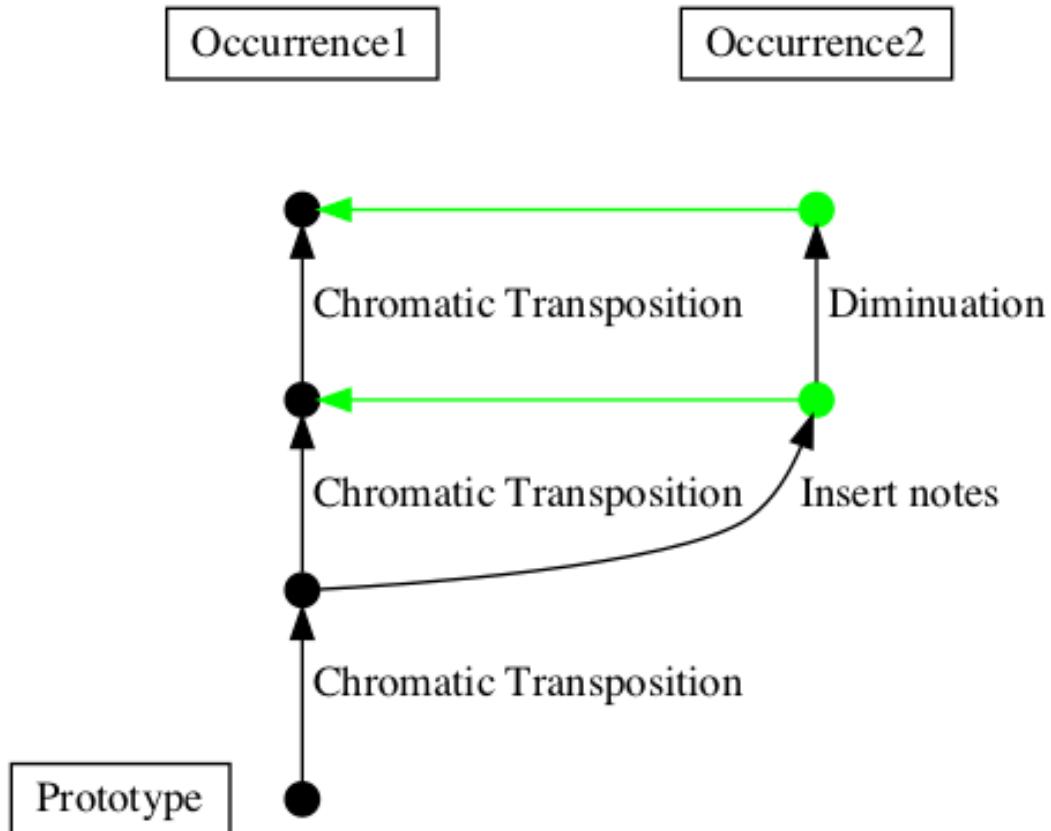


Figure 5.7: An example using transformations to compare two pattern occurrences.

& Ghifary, 2016), the authors hypothesised that strong-typing will provide a solid foundation for algebraic operations on learned features and that "strong-typing may then provide a useful organising principle in future machine reasoning systems".

Compositionality in ML is also important, but in a slightly different sense than FP, as the learned functions are seen as a composition of simpler operations. As reported in some models, from the initial layer to the last layer in a neural network, we see a hierarchy of the features going from initially low-level, then mid-level, and finally high-level features that can enable better generalisation. For example, a semantic hierarchy emerges in Style-Gan, going from layout, objects, attributes, colour schemes. (Yang *et al.*, 2019) The compositionality, how these features integrate with each other, is core to the multiple layers in deep learning methods.

As another connection between FP and ML, [Programming Language Processing \(PLP\)](#) has given back to [PL](#) by providing functionality and tools such as bug finding, code2vec (learned model of code representation), AnyCodeGen (Probabilistic code generation), and Tabnine (AI code completions). How is code different from text and music after all?

The little importance of language choice?

We argued that functional programming has many advantages for modelling patterns. The dream is that one can build and modify their tools in an ergonomic way with the appropriate DSL. The willingness of a domain expert to learn a new tool or language, however, seems to be a mix of enthusiasm and reluctance in reality (Collins, 2011):

Perhaps it is optimistic to expect music analysts to welcome or employ such a discovery tool, especially if it requires learning a command-line language, but Huron (2001b) demonstrates effective use of a pattern matching tool in preparing a music-analytic essay.

In many cases, the ideas behind the actual programming language or DSL are perhaps more important than the technologies themselves. For example, recall in Chapter 4, one such idea is the use of synthetic data as a testing structure. They were actually created with Haskell and the QuickCheck package, known as a package for the idea of property testing. In addition to this idea, by employing the idea of laziness in Haskell, we can easily change the length of the patterns for different purposes.

The same data can be generated using a variety of other programming languages, just as QuickCheck has implantations in a variety of programming languages. We can, however, benefit from using Haskell by connecting between the ideas of augmenting data, property testing, and laziness across different fields. There is also the added benefit that the control and structure of the program would be clearer and more ergonomic.

Transformations and the unique musicality of humans

To expand the discussion on the perception of transformations, we can make a connection to neuroscience, where the unique traits of human musicality is studied. Musicality refers to the cognitive and neural mechanisms that underpin basic musical behaviours that often do not require formal training to develop (Honing *et al.*, 2015), and therefore has a close connection with the common transformations we mentioned in this chapter. Examples of musicality include recognising transposed melodies or tapping along with rhythms. Musicality can also be compared to the language capacity suggested by (Chomsky, 1965): humans seem to have unique cognitive infrastructures underlying language acquisition. Some transformations, repetitions and variations, and patterns are so universal that the brain may have overlapping areas for them (Patel, 2003). These transformations appear to be unique to

5 Computation, Functional Programming, and Music

humans. By contrast, songbirds, for example, can learn human melodies but do not recognise them when transposed (Bregman *et al.*, 2016).

Chapter 6 Modelling Patterns with Transformations using Haskell

A language that doesn't affect the way you think about programming is not worth knowing.

– Alan J. Perlis

Programming in different languages is like composing pieces in different keys, particularly if you work at the keyboard. If you have learned or written pieces in many keys, each key will have its own special emotional aura.

– Douglas Hofstadter

With previous chapters, we stand on the following grounds:

- Evaluating musical pattern discovery algorithms is a difficult problem
- Musical patterns, transformations, and Haskell have important connections

The aim of this chapter is to introduce the Haskell implementation for our proposal to use transformations to model patterns. We start with a broad overview, reiterating important arguments from previous chapters for our design choices. We give a full list of transformations we will be using for our implementation. The majority of this chapter will be dedicated to explaining the inner workings of Pattrans, a DSL we implemented for exploring the connection between musical patterns and transformations. We end the chapter with a summary and discussion of limitations and future work.

6.1 Overview

In the previous chapter, we examined a few works on patterns and transformations, and the strong connection between Haskell and music applications. By relating pattern occurrences via transformations, we may gain a better understanding of what

kinds of patterns are extracted by these algorithms and how they compare to human annotated patterns. Building on top of the groundwork laid in previous chapters, we motivate this connection once more but with condensed arguments and two examples.

Musical patterns are closely related to repetition and variation. On the one hand, repetition and variation of a musical patterns have been used and can be perceived in a lot of musical material. Conversely, patterns can be defined as the repetition and variations of an initial musical element.

Independent of the variation, there must be some original material from which the variations stem. We refer to these as the *prototype patterns*. For now, we take the first occurrences of patterns as the prototype of a pattern. Each prototype pattern may be related to several pattern occurrences. Typically, there is some transformation between the prototype pattern to other pattern occurrences, accounting for the variation amongst pattern occurrences.

Unsurprisingly, we can model these transformations as functions. For example, one simple transformation is the chromatic transposition, $\lambda x \rightarrow x + n$, that transposes each note by some pitch shift n . In Figure 6.1, we come back to the example of a simple pattern and its occurrences in an étude. The same bar of notes is repeated successively, shifted upwards or downwards in the next occurrence: a pattern emerges—the occurrences are related to each other via the transposition in pitch. The second occurrence can be viewed as a variation of the first occurrence, realised by the transformation of transposition. More transformations are known to exist in music theory and can be used to characterise different types of repetitions with variations. Similarly, Figure 6.2 shows pattern occurrences annotated by musicologists with a diverse range of transformations. Because many transformations only differ by a couple of notes, we need to consider the local type of transformations—approximations.

There are other properties that can be explored using the formulation of transformations, such as the equivalences between compositions of transformations, the transformations of transformations, and so on. A group of occurrences of the same pattern can even be seen as an equivalence class, tied together by transformations, but we need to be careful because some of the transformations or their combinations might not be transitive, as discussed in Chapter 2. In such a rich topic, in this dissertation, we have three specific reasons for looking into transformations:

- Relevance to algorithms: many general and musical pattern discovery algorithms either explicitly or implicitly consider these transformations important.

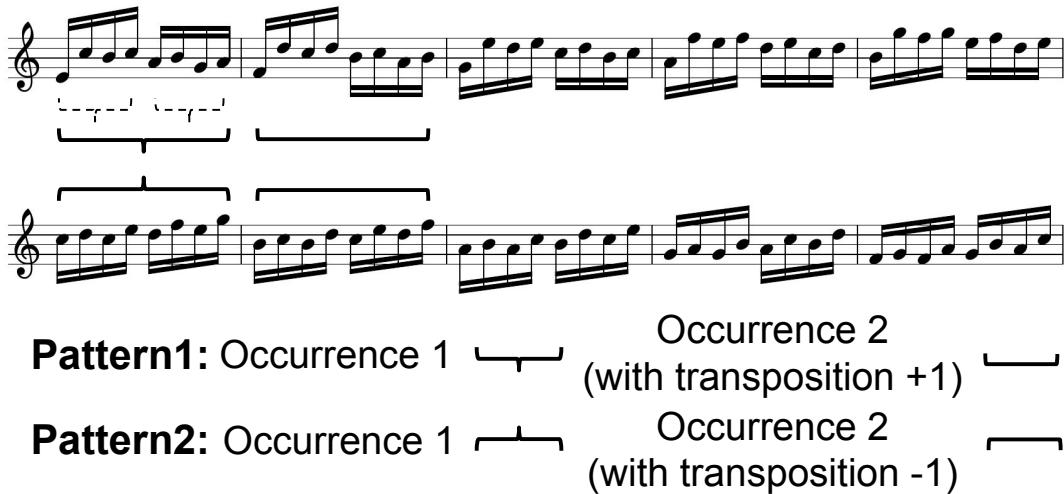


Figure 6.1: Musical pattern and transformation examples in simple études. The patterns and occurrences are marked with over-/under-braces. We only consider a two-level hierarchy of pattern-occurrence; sub-patterns as in the dotted braces are not considered.

- Relevance to music analysis: transformations have been widely used in musical composition and analysis, often in combination with mathematical theories such as set theory and category theory.
- Relevance to functional programming: to encode and combine transformations, automate the checking and discovery of transformational relations of musical patterns, functional programming approaches are very suitable.

We hence propose to model musical patterns by modelling the repetition and variation amongst pattern occurrences. Each transformation essentially relates two occurrences of the same pattern. A pattern, in this sense, is a name given to a set of occurrences. When we take the first occurrence of the pattern as its prototype and match the rest of the occurrences with it, we obtain a set of pattern occurrence relations. What distinguishes a set from the other, in our case, then is the different compositions of relations between the occurrences. In Figure 6.3, we show this schematically.

This chapter

In the rest of this chapter, we first detail the transformations we focus on for this chapter and the next. We then describe the Haskell implementation of the transformation-based model for musical patterns, in the form of an embedded [DSL](#), Pattrans. We demonstrate this [DSL](#) in three different capacities in the next chapter.

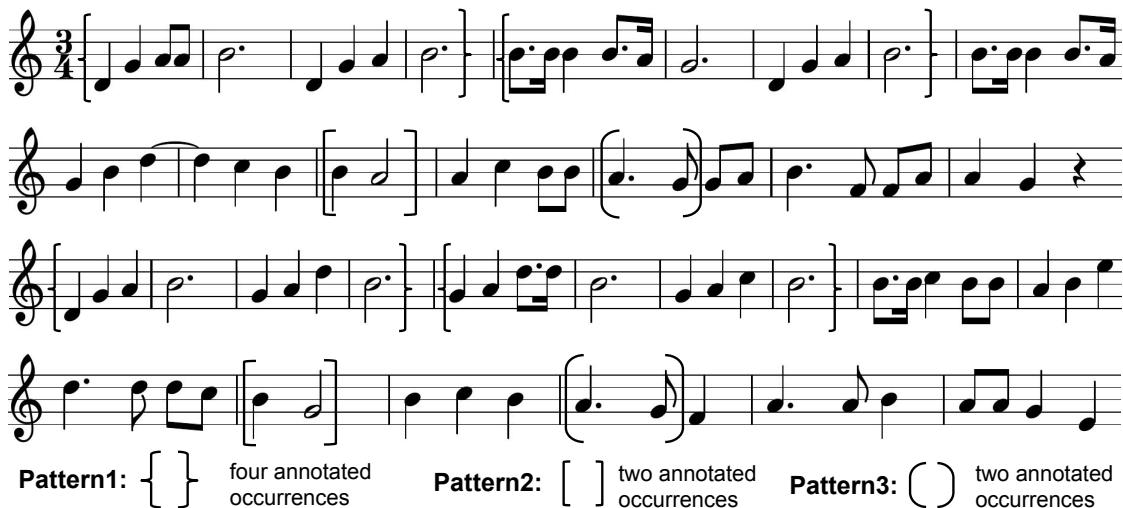


Figure 6.2: Musical pattern and transformation examples in MTCANN. There are some exact repetitions. The local type of transformations—approximation—is also needed to describe the relations between the pattern occurrences.

6.2 Musical transformations in Pattrans

Before we go into the implementation, we describe and explain the transformations we are going to use in Pattrans. From a range of literature in Chapter 5, we identified the following important transformations that apply to all notes in a musical pattern:

- Shifting
 - in time: *exact repetition*
 - in pitch: transposition, comes in different flavours (*chromatic transposition, tonal transposition*)
- Reflecting:
 - in time: *retrograde*
 - in pitch: *inversion*, similar to transposition that it comes in different flavours
 - composition: retrograde-inversion
- Scaling:
 - in time (*augmentation, diminution*)
 - in pitch (many formulations are possible; not considered in detail in this dissertation)

In addition to these, we also identified the importance of approximations—the local transformations that do not apply to all notes in a pattern, but a locally a few,

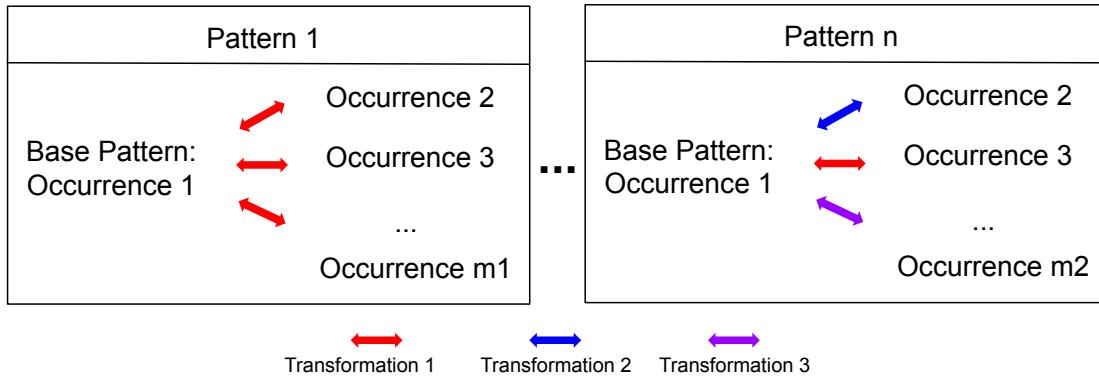


Figure 6.3: Pattern occurrences connected by transformations. We propose extracting the different transformations—the different colours of arrows in the figure—to form the basis for comparing different patterns. The composition of the different colours of arrows can be what distinguishes one pattern from another.

such as the insertion or deletion of certain notes. In the following of this section, we will introduce different groups of transformations we implement in Pattrans.

Primary and miscellaneous transformations

From the lists of transformations we have seen so far, we take three prominent transformations in particular as our primary transformations:

- **Exact repetition**: repeat an occurrence with exactly the same musical events. That is, horizontal translation preserving rhythm and pitch.
- **Chromatic transposition**: move pitches by a fixed number. That is, vertical translation preserving rhythm.
- **Rhythmic-only repetition**: repeat the rhythm of an occurrence while permitting any pitch transformations other than the two considered above.

The rhythmic-only repetition in this list has not received much discussion yet. We use this transformation partially due to the lack of transformations in scaling pitch, partially due to the influence of the feature analysis in Chapter 4, where rhythmic features were important for distinguishing between algorithmic and human-annotated patterns.

We also consider a miscellaneous (misc.) transformation group, which contains the following transformations and their compositions: retrograde, augmentation, diminution, and approximation. Other potential non-primary, non-misc. transformations and relations are referred to as "unclassified" (so named because we cannot find them yet with the transformations above).

Formally speaking, we have a list of musical events in the prototype pattern occurrence, $P = \{(t_i, p_i)\}, i \in \mathbb{N}$ (hereon, we assume i , the indices of musical events, are natural numbers), and $P' = \{(t'_i, p'_i)\}$ the list of musical events in the pattern occurrence to compare it with, where t_i and t'_i are the onset sequences of the pattern occurrences, and p_i and p'_i the corresponding pitch sequences. Let the **Inter-Onset Intervals (IOIs)** sequences be $\Delta t_i := t_{i+1} - t_i$ and $\Delta t'_i := t'_{i+1} - t'_i$. Note that, in this formulation, we do not consider the cases where P and P' are of different lengths, because the length difference breaks the criteria of being under one of the three primary transformations we consider. The checking criteria for the three primary transformations correspond to:

- (1) **Exact repetition** $(p_i = p'_i) \wedge (\Delta t_i = \Delta t'_i), \forall i$
- (2) **Chromatic transposition** $\exists! c \in \mathbb{N}, \text{s.t.} (p_i = p'_i + c) \wedge (\Delta t_i = \Delta t'_i), \forall i$
- (3) **Rhythmic-only repetition**: $(\Delta t_i = \Delta t'_i), \forall i$

Intuitively, there is a gradually loosening process between the three transformations on how strictly the pitches need to match. Exact repetition requires pitch to be preserved exactly; chromatic transposition requires a constant pitch offset value; and rhythmic-only repetition does not impose any constraints on pitches. Furthermore, the last transformation covers the cases of other common music transformations: tonal transposition and inversion. We do not emphasise tonal transposition in our list of transformations because it is not always possible to compute the one-and-only correct key (“2005:Symbolic Key Finding Results - MIREX Wiki”, 2005) and then find the appropriate tonal transposition given a short pattern occurrence. These transformations have also been shown to be perceptually relevant for listeners under certain circumstances according to the discussion in Section 5.3.5 and discussions in (Krumhansl *et al.*, 1987; Meredith *et al.*, 2002; Temperley, 1995; Thorpe *et al.*, 2012), despite ongoing research and disagreement on the generalisability of such perceptibility. For the above reasons, we group transformations into the three primary transformations and misc. transformations. Although it is possible to change this grouping in Pattrans as we will show in Section 6.3.2, this grouping will remain important in the next chapter where we present the analytical results from Pattrans.

Though this list is not comprehensive, by using well-established and not over-complicated transformations, we equip ourselves with the power to examine musical patterns rigorously and quantitatively. For example, we can compute how many occurrences of a pattern are related to a prototype occurrence via the transformations, and what kind of transformations can be observed in a set of occurrences.

Such data can be used to investigate different forms of musical patterns. This computation is what we will do in the next chapter.

Approximation

The approximations in the misc. transformations deserve an extra mention here for clarity. As we discussed before, we extend the primary transformations by allowing the occurrences to be related in an approximate manner. Primary transformations transform every musical event in the occurrence under consideration. This global type of transformation provides a concise way to describe the changes of individual musical events in bulk by globally including all musical events in a pattern.

The limitation is that when there are smaller variations, e.g. the insertion of a grace note, primary transformations will not be able to capture this fine-grained similarity. For example, measures 1-2 and measures 3-4 in Figure 6.2 only differ by one note. Therefore, we also consider the more ad-hoc insertion and deletion as the transformation of "approximation" on top of the primary transformations. This is to account for the cases where a small number of notes are changed in the context of a transformation of a larger group of notes. For example, imagine the sequence of pitches C, D, E, F, G, along with the exact repetition and one note approximation; it can be related to the pitch sequence C, D, E, F, F. This idea has obvious connections with the notion of similarity and edit distance. We will show how we implemented this in Pattrans in Section 6.3.5. For the moment, we focus on this functionality to simulate adding, deleting, and modifying a few notes.

Other groups of transformations

In (Melkonian *et al.*, 2019), the paper on which this chapter is based, we chose a group of transformations as atomic transformations primarily due to their simplicity: [exact repetition](#), [chromatic transposition](#), [inversion](#), [retrograde](#), [augmentation](#), [diminution](#), [retrograde inversion](#), [rotation](#). In this dissertation, we have re-formulated this into primary transformations and misc. transformations, as we will be using them in the next chapter.

Orders of transformations

When considering computationally implementing multiple transformations, the issue of order arises: given two transformations t_1 and t_2 , which one should be checked first? One can propose checking all transformations and their combinations against every pair of musical patterns we will be comparing with. This brute-force

approach can be very computationally expensive. However, as we mentioned earlier in this section, some transformations are stricter than others, so we can check the strictest transformation first, and then the next strict one for the patterns that have not been paired. In a similar vein, we check global transformations first in the case of global and local transformations. This strict-to-loose and global-to-local approach is exactly what we do in Pattrans.

6.3 Pattrans

Pattrans¹ have the following decoupled components: types (Section 6.3.1), transformations (Section 6.3.2), discovery (Section 7.2), and other technical components such as import, rendering, and MIDI that we will not cover in this dissertation. In the following, we will overview the main techniques we used to implement our meta-analytic framework. Although we chose Haskell for our implementation, we do not rely on any of Haskell's advanced type-level features (e.g. data kinds); thus any other strongly-typed functional programming language would suffice.

6.3.1 Basic types

According to (Brady, 2017), "In type-driven development, we use types as a tool for constructing programs. We put the type first, treating it as a plan for a program and use the compiler and type-checker as our assistant, guiding us to a complete and working program that satisfies the type." Starting from the fundamental elements in music, we define the types of time and pitch as in Code Snippet 6.1. In our encoding, we used the simplest possible representation of musical events. Time is represented as the number of crochet beats from the start of the song, while MIDI numbers represent pitch, in the form of plain integers.² Other primitives include scale degree (integer), interval (integer), octave (integer)³.

We then assemble these basic types to create a basic musical event: a note, which consists of a time and pitch value. A pattern (occurrence) is a group of notes. Both types are shown in Code Snippet 6.2.

According to the output of many pattern discovery algorithms, we define a set of occurrences to be a pattern (group). In each pattern group, we have a prototype

¹The implementation is available at <https://github.com/omelkonian/hs-pattrans>

²Here, we only consider the part of a MIDI value that represents pitch number, ignoring other features such as velocity.

³Not all codes can be found in the main text of this dissertation. More code can be found in Appendix app:code.

```
-- | Time in crochet beats.
type Time = Double
-- | MIDI values are represented with integers.
type MIDI = Integer
-- | Intervals are represented with integers (i.e. number of
--   ↳ semitones).
type Interval = Integer
```

Code Snippet 6.1: Basic types for defining musical concepts.

```
-- | A pattern is a sequence of notes.
type Pattern = [Note]

-- | A simplistic music note (only time and pitch).
data Note = Note { ontime :: Time -- onset time
                  , midi :: MIDI -- MIDI number
                  } deriving (Eq, Show)
```

Code Snippet 6.2: Define pattern (occurrence) and note.

pattern and (possibly many) occurrences of that pattern across the musical piece. We also keep some metadata (such as to which piece of music a pattern group belongs), which can be used later in the analysis. The definition of the pattern group in Haskell can be seen in Code Snippet 6.3.

```
-- | A pattern group is one of the patterns of a piece of music,
--   ↳ identified by an expert or an algorithm, and defined by a
--   ↳ pattern prototype and other pattern occurrences.
data PatternGroup = PatternGroup
  { piece_name :: String
  -- ^ the name of the music piece, that the pattern group belongs
  --   ↳ to
  , expert_name :: String
  -- ^ music expert or algorithm that produced the pattern
  --   ↳ occurrences
  , pattern_name :: String
  -- ^ the name of the current pattern group
  , basePattern :: Pattern
  -- ^ the pattern prototype (always taken from the first
  --   ↳ occurrence)
  , patterns :: [Pattern]
  -- ^ all other pattern occurrences of the prototype
  } deriving (Eq)
```

Code Snippet 6.3: A group of pattern (occurrences) becomes a pattern (group).

Lastly, certain properties of music elements can be useful when we later define transformations. For instance, for transpositions, we might want to see the intervals between a pattern's pitch values, rather than the absolute pitches in isolation. To convert between different types, for example to extract durations and intervals from patterns, we can define functions which take an input of type **Pattern** and compute a list of **Time** or **MIDI**, as shown in Code Snippet 6.4.

```

toRelative :: Num a => [a] -> [a]
toRelative = fmap (-) . pairs

pitch, intervals :: Pattern -> [MIDI]
pitch      = fmap midi
intervals  = toRelative . midi

durations :: Pattern -> [Time]
durations = fmap (uncurry (-)) . pairs . onsets

```

Code Snippet 6.4: Extraction of intervals and durations.

6.3.2 Transformation checking

We have specified the transformations we will be implementing in Section 6.2. In Pattrans, we have constructed the DSL so that one pick their own sets of transformations with ease. For example, in Code Snippet 6.5, we define a default analysis with four transformations (exact repetition, transposed, inverted, and retrograded) and an analysis called "exact" that only considers exact repetitions. Both of them are defined along with different degrees of approximation (1, 0.8, 0.6, 0, 4, 0.2).

```

analyses :: [(String, Analysis)]
analyses =
  [ ( "default"
    , ( [ ("exact",          (exactOf ~~))
        , ("transposed",     (transpositionOf ~~))
        , ("inverted",       (inversionOf ~~))
        , ("retrograded",    (retrogradeOf ~~))
        ], [1,0.8..0.2] ) )
  , ( "exact"
    , ( [ ("exact", (exactOf ~~))
        ], [1,0.8..0.2] ) )

```

Code Snippet 6.5: One can define different sets of transformations for analysis. The transformations are then checked one by one in order.

Now let us consider how we can implement these analyses we want. The first thing to realise is that the definitions of our transformations will essentially be a predicate on two elements, returning `True` when they can be related by a transformation and `False` otherwise. Therefore, we define a `Checker` type as in Code Snippet 6.6. We provide a convenient infix notation to check for this, as well as a type alias for *homogeneous* checkers of elements of the same type.

An alternative way to define this is shown in Code Snippet 6.7 and is used in our library. The definition in Code Snippet 6.6 is more readable but less expressive than 6.7. We will use the style in Code Snippet 6.6 in this dissertation.

Once we have the `Checker` type, the simplest possible comparison is that of exact equality between elements of the same type, which can be defined using the `Eq` typeclass, as shown in Code Snippet 6.8.

```

data Check a b = MkCheck (a -> b -> Bool)
(<=>) :: a -> b -> Check a b -> Bool
(x <=> y) (MkCheck p) = p x y

```

```
type HomCheck a = Check a a
```

Code Snippet 6.6: Define checker for predicating on two elements..

```

newtype Check a b = Check { unCheck :: a -> b -> Bool }
(<=>) :: a -> b -> Check a b -> Bool
(x <=> y) p = unCheck p x y

```

Code Snippet 6.7: A different way to define checker.

Eventually, we would like to write checkers for exact repetition as shown in Code Snippet 6.9. To be able to achieve this, we need compositional operations to glue together the different features (e.g. pitch and rhythm) with the checker. In addition, we can extend the notion of equality in the checker to check the approximation of a pattern. As an example of all these combinations, in Code Snippet 6.10, we can write a checker that checks inversions with different degrees of approximation. In Code Snippet 6.11, similarly, we show the implementations of the two other primary transformations. In the following sections, we will detail the compositional operators and the implementation of approximations.

6.3.3 Compositions of checking transformation

In order to combine multiple checks in conjunction, we must recognise the monoidal algebraic structure of checkers. Note that **Check** can be seen as a Monoid in more than one way, i.e. using either conjunction or disjunction. Here we stick to the conjunctive version (note the use of `&&`), since we do not have any use-cases for disjunctive checkers yet. If such a DSL construct is needed in the future, we will provide newtype wrappers and the programmer would then need to manually annotate which monoid instance to use (when one uses `<>`). The implementation of the monoidal structure can be seen in Code Snippet 6.12.

In order to define more interesting pattern relations with compositional operations, we first make the observation that the type parameters (a and b in this case)

```

-- e.g. ([1,2] <=> [1,2]) -> True
-- e.g. ([1,1,2] <=> [1,2,1]) -> False
equal :: Eq a => HomCheck a
equal = Check (==)

```

Code Snippet 6.8: The simplest possible HomCheck.

```

exactRepetitionOf :: HomCheck Pattern
exactRepetitionOf = rhythm >$< equal
                  <> pitch >$< equal

pitch :: Pattern -> [MIDI]
pitch = fmap midi

durations :: Pattern -> [Time]
durations = fmap (uncurry (-)) . pairs . onsets

rhythm :: Pattern -> [Time]
rhythm = map (truncate' 2) . durations

-- helper function to truncate long decimal duration value to two
-- digits
truncate' :: Int -> Double -> Double
truncate' n x = fromIntegral (floor (x * t)) / t
  where t = 10^n

```

Code Snippet 6.9: Checker for exact repetition and the types of its components.

```

inversionOf :: ApproxCheck Pattern
inversionOf = basePitch >$< equal
                  <> rhythm >$< approxEq2
                  <> intervals >$< (inverse &gt; approxEq2)

```

Code Snippet 6.10: Checker for inversion up to approximation.

occur in a *contravariant* position, since they are arguments to a function. Moreover, we observe that there are two different arguments and we want functorial action in both. As an example of contravariance, assume you have a checker for durations (i.e. `HomCheck Time`) and you want to use that to check the equivalence of two patterns (i.e. `HomCheck Pattern`). Hence, you need to have a function `HomCheck Time -> HomCheck Pattern`, but you cannot define it even if you have a function `Time -> Pattern`. What you, in fact, must have in your hands is a function `Pattern -> Time` (notice the reversal in the argument order, namely, *contravariance*). With these observations, we arrive at the concept of a *contravariant bifunctor* and define the instance for our checker type, as shown in Code Snippet 6.13.

We also define specialised infix variants of the contravariant operations on checkers of equal types, where we modify one or both of the arguments, as shown in Code Snippet 6.14. Figure 6.4 depicts the associated contravariant operations diagrammat-

```

transpositionOf :: ApproxCheck Pattern
transpositionOf = rhythm >$< approxEq2
                  <> intervals >$< approxEq2

rhythmicOnly :: ApproxCheck Pattern
rhythmicOnly = rhythm >$< approxEq2

```

Code Snippet 6.11: Checker for chromatic transposition and rhythmic-only repetitions.

```

instance Monoid (Check a b) where
  mempty = MkCheck (\_ _ -> True)
  p <> q = MkCheck (\_ x y -> (x <=> y) p
                      && (x <=> y) q)

```

Code Snippet 6.12: Implementation of the monoidal structure of checkers.

```

class ContravariantBifunctor p where
  contraBimap f g = contraFirst f . contraSecond g
  contraFirst f = contraBimap f id
  contraSecond g = contraBimap id g
instance ContravariantBifunctor Check where
  contraBimap f g p = Check \$ \ x y -> (f x <=> g y) p

```

Code Snippet 6.13: Definition of contravariant bifunctor.

ically for homogeneous checker, assuming $f :: b -> a$ and $\eta :: a -> a$. The most general case where $f :: c -> a$ and $g :: d -> b$ is given in Appendix C.

By having the operators for composition in place, we can decouple many features of patterns, such as time and pitch, into their individual elements. The transformations also become easy to control, combine, and consider together with these features.

```

(>$<) :: (b -> a) -> HomCheck a -> HomCheck b
f >$< p = contraBimap f f p

(>$) :: (a -> a) -> HomCheck a -> HomCheck a
(>$) = contra1

($<) :: (a -> a) -> HomCheck a -> HomCheck a
($<) = contra2

```

Code Snippet 6.14: Infix variants of the contravariant operations on checkers of equal types.

6.3.4 An excursion into category theory

With the concept of contravariant bifunctor, together with monoids, we enter the territory of category theory. Category theory has been used to model phenomena in many areas we have discussed in previous chapters, such as cognitive science, machine learning, and functional programming (Arjonilla & Ogata, 2017; Fong *et al.*, 2019). The central ideas are objects and morphisms, which can be intuitively understood as states and the processes or connections between states. This generalised formulation gives rise to an extensive network of analogies and relationships between

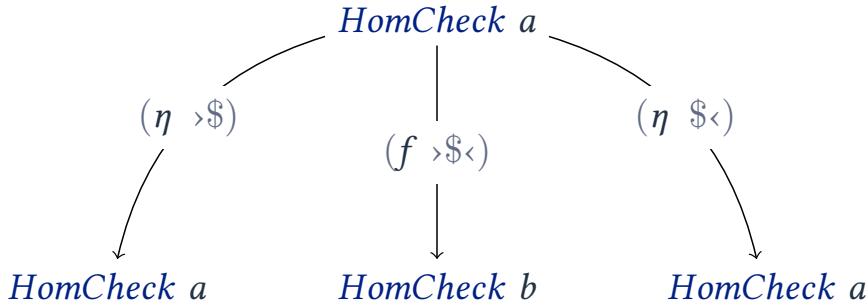


Figure 6.4: The contravariant bifunctor structure of HomCheck.

fundamental subjects such as physics, topology, logic, and computation (Baez & Stay, 2010).

In Appendix C, we give the formal mathematical definition of a contravariant bifunctor. For a more comprehensive introduction, please refer to other resources such as (Milewski & Tabachnik, 2018). In the remainder of this section, we will mention some more applications of category theory in related domains, namely, Haskell and music.

In the domain of Haskell, profunctor optics (Pickering *et al.*, 2017) are used to present a framework for modular data access. The definition of profunctors, as shown in Code Snippet 6.15, is very similar to that of our contravariant bifunctor, except that one variable is in a covariant position.

```

class Profunctor p where
  dimap :: (a' -> a) -> (b -> b') -> p a b -> p a' b'

class ContravariantBifunctor p where
  contraBimap :: (a' -> a) -> (b' -> b) -> p a b -> p a' b'
  
```

Code Snippet 6.15: The definition of profunctor and contravariant bifunctor.

Category theory also has tight connections with type theory. There are many benefits to a program standing on the solid foundation of a mathematical model. For example, this foundation can provide a common language and assist with deducing properties of the model.

Category theory has also been used to model music structure. For example, in extending Lewin's transformational networks, (Rahn, 2004) provides the insight that transformational networks are essentially graphs which have arrows labeled in some semigroup, and nodes in some set acted on by the semigroup. In this way, the network can be extended in category theory. More work, such as (Andreatta, 2018; Giesa *et al.*, 2011; Mannone, 2018), have used category theory to model a variety of musical activities and made interdisciplinary connections.

6.3.5 Approximation

In this section, we come back to the implementation of Pattrans, and more specifically, the approximations. As mentioned before, we aim to model local deformations or changes in the music with approximation. Although the primary transformations and their composition can account for a considerable number of occurrence relations, they can be limited in the presence of variations such as embellishment or ornamentation. We therefore add more flexibility by allowing inexact matches, essentially allowing the insertion and deletion of musical events.

This is not an easy problem to solve, and can be connected with other issues and concepts such as similarity and distances. In our implementation, we choose the best way we can to represent our intuition on this matter: to allow different degrees of approximation, we specify percentages for how many insertions and deletions we allow when comparing a pair of musical patterns. For example, we can match a list of pitches, $[A, C, F, A, B]$, and $[A, C, G, A, B]$ with an approximation of 80%. Formally, we extend the criteria for a "match" with the approximation of percentage p : when at least p of the prototype is in the occurrence, i.e.

$$\text{ignored} \leq (1 - p) * N \quad (6.1)$$

and when at least p of the occurrence is in the prototype, i.e.

$$\text{added} \leq (1 - p) * M \quad (6.2)$$

where N, M , are the lengths of the prototype and occurrence, respectively; added is the number of elements in the occurrence that do not appear in the pattern, ignored is the number of prototype elements that were not deleted i.e. $M - (N - \text{added})$.

In the example illustrated above, we have $N = M = 5$ and we *ignored* the prototype note F and *added* the occurrence note G (i.e. $\text{ignored} = 1$ and $\text{added} = 1$). Thus we can conclude that these two patterns are 80% equal, since the two required conditions are satisfied:

$$\text{ignored} = 1 \leq 1 = (1 - 0.8) * 5$$

and

$$\text{added} = 1 \leq 1 = (1 - 0.8) * 5$$

.

In implementation, we first define this approximate checker type by passing a floating point number $p \in [0, 1]$, representing the approximation degree as a percentage, as shown in Code Snippet 6.16. We then approach this approximate checking in two steps. First, we define a function that gradually deletes elements from a list, given that certain criteria are satisfied. The process and intuition behind the deletion criteria is rather simple; for each element of the prototype:

- Traverse the occurrence until the prototype element is found.
- If found, consider two cases:
 - Increase added by the number of bypassed elements during the search.
 - Ignore it nonetheless, incrementing ignored.
- Otherwise, increment ignored.
- Continue with the rest of the prototype.

In this way, we can then compute "ignored" and "added" values and compare them with the p value, the approximation degree that we desire, and finally obtain the results True or False from this checker, as shown in Code Snippet 6.17.

```
type ApproxCheck a = Float -> HomCheck a
(~~) :: ApproxCheck a -> Float -> HomCheck a
approxChecker ~~ p = approxChecker p
```

Code Snippet 6.16: Definition of approximation type and function.

```
approxEq :: Eq a => ApproxCheck [a]
approxEq p = Check $ \xs ys ->
    (ignored <= (1 - p) * length xs)
    && (added <= (1 - p) * length ys)
where
    (added, ignored) = del ys xs

del :: [a] -> [a] -> (Int, Int)
del ys []      = (length ys, 0)
del [] xs     = (0, length xs)
del ys (x:xs)
    | Just (ysL, ysR) <- x `del1` ys
    = (first (+ length ysL) <$$> del ysR xs)
        `min` (second (+ 1) <$$> del ys xs)
    | otherwise
    = second (+ 1) <$$> del ys xs
```

Code Snippet 6.17: Function that search for approximations..

The worst-case time complexity of the algorithm above is $O(NM)$ where N, M are the lengths of the prototype and occurrence respectively, since we would need to traverse the whole occurrence for each element of the prototype. Nevertheless, we set a maximum look-ahead on the deletion process to facilitate faster analyses, resulting

in overall runtime complexity of $O(N)$, since we would perform $O(1)$ computation for each of the N elements of the prototype⁴.

When we consider the values computed from pairs of basic elements (e.g. intervals from pitch), they score rather badly on the previous definition of approximate equality, since a simple insertion on the initial pattern would create a lot of differences in the paired output. As a motivating example, assume our prototype consists of the notes A and B and the occurrence adds a passing note B^b in-between them. While first-order approximate equality works well when comparing their pitch, we get in trouble when comparing their intervals; we have lost all similarity between them, since the prototype has intervals [2] and the occurrence [1, 1].

We, therefore, consider two types of approximation, and call the one we just covered as first-order and move on to cover the second-order approximation. We call what in Code Snippet 6.17 first-order because we only compare the sameness of the token. What we call second-order approximation also considers the intervalic equivalence between two tokens. We define it in such a way that we additionally allow the merging of consecutive elements in the occurrence and match them to a single entity of the prototype. Code Snippet 6.18 shows the type signature of the second-order approximation, and we show the rest of the code more extensively in Appendix D.

```
approxEq2 :: (Show a, Ord a, Num a, Eq a) => ApproxCheck [a]
approxEq2 p = ...
```

Code Snippet 6.18: The type signature of the function that implements second-order approximation.

To give an example, for two numerical lists and checking whether they are 75% approximate to each other:

```
([2,1,2,2] <=> [1,1,3,2,1,1]) (approxEq2 ~~ 0.75)
```

Second-order approximate equality allows us to merge the first two and the last two intervals of the occurrence, replacing them with their sum:

```
([2,1,2,2] <=> [2,3,2,2]) (approxEq2 ~~ 0.75)
```

We now have only a single ignored interval in the prototype and a single added interval in the occurrence, thus it is safe to conclude that these (interval) patterns are 75% equal:

$$\text{ignored} = 1 \leq 1.0 = (1 - 0.75) * 4$$

$$\text{added} = 1 \leq 1.5 = (1 - 0.75) * 6$$

⁴Note that the faster implementation is actually an under-approximation of the proposed algorithm, i.e. there might be false negatives in the results.

To use this package in our experiments in the next chapter, we perform multiple passes of comparisons by varying the approximation degree of 100% (exact repetition), 80%, 60%. We note the approximated transformations by appending a single-digit number at the end of the names of the transformations and their combinations. For example, `exact8` denotes a match of exact repetition but with at most 80% approximation, i.e. setting $p = 0.8$ in the previous equations.

6.4 Summary

In this chapter, we presented an expressive DSL to describe (monophonic) music-theoretic transformations, based on simple notions of category theory, namely monoids and contravariant functors. The transformations we used were explained and motivated. To accompany the global transformations, we also discussed and instantiated the generic notion of approximate equality. We propose a complete quadratic algorithm to perform this approximation, but also provide a linear variant that is faster under-approximation that we use to produce our results.

One of the significant benefits of the transformation approach is its compositionality, meaning we can express meaningful transformations in terms of simpler ones. The functional programming language Haskell also gives us advantageous controls and architecture of the program. We summarise the groups of transformations and their combinations in Figure 6.5.

6.5 Discussion

We do not claim that our model is comprehensive. We certainly do not touch upon the intentionality behind the musical transformations, i.e. determining whether a musical transformation was applied as a technique or accidental, which has a separate level of intricacy in itself. Instead, we merely use transformations to describe computations that can be used to systematically and quantitatively probe the musical patterns given by human annotators and algorithms. Using this DSL, we will see what kind of insights can be gained in the next chapter. In the remainder of this section, we will discuss some limitations of our implementation.

Multiple transformations

When t_1 and t_2 can both be used to match two pattern occurrences, we can take a pluralist view (taking both t_1 and t_2 as valid matches) or consider extra factors that

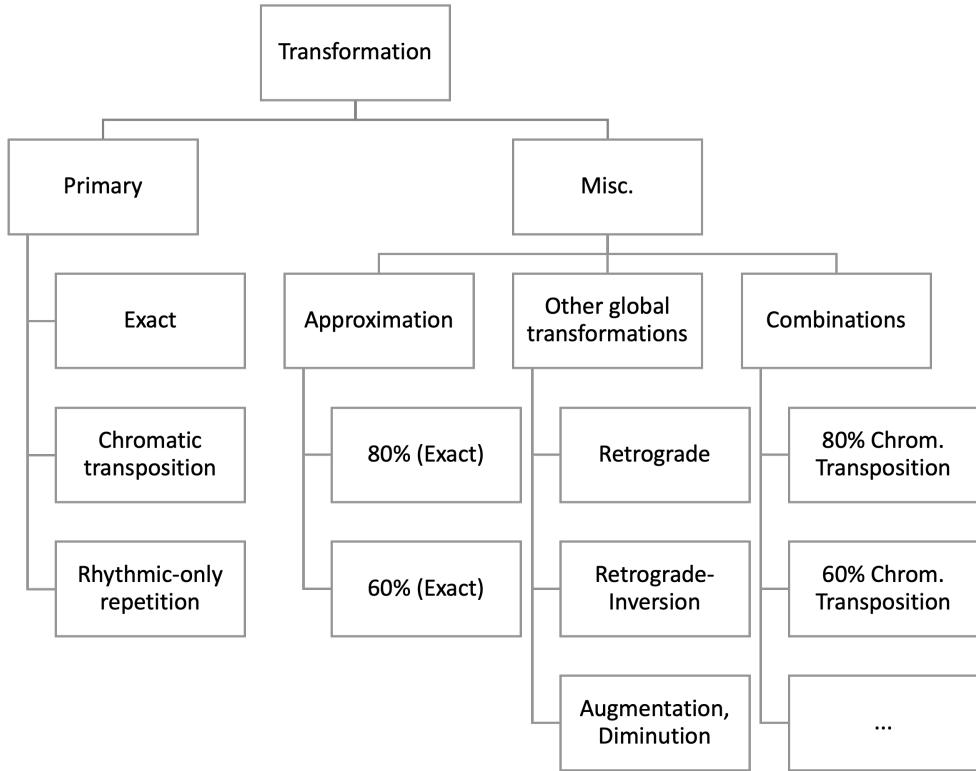


Figure 6.5: The primary and miscellaneous groups of transformations we implement in Pattrans. Chrom. is short for chromatic transposition. We do not list all the combinations such as 80% augmentation.

could be used to rank transformations, thereby enabling us to choose between t_1 and t_2 . In Pattrans, we take the transformation in a certain order (See Section 6.2) .

If we allow ourselves to multiple matching and match pattern occurrences with the compositions of transformations, e.g. t_1 (t_2 (t_3)), we also need to additionally consider properties of these compositions such as associativity, commutativity, and potential equivalences between them. These are the situations we are not considering for this dissertation.

Prototype patterns

In Pattrans, we choose the first pattern that occurred in time to be the prototype occurrence. This is not an unfounded choice: as we discussed in Chapter 2, in music, the patterns that appear earlier in time in a piece have more chance to be repeated and varied in the rest of the piece. A temporal discount factor seems to be applicable when considering how likely a musical excerpt can be a prototype pattern.

This is not always the case, however, and selecting a prototype pattern is a difficult problem: one must know which occurrence in a set is the prototype in order to test the membership of other candidate occurrences, but one must consider all

occurrences in order to determine which is the prototype. In addition to treating the first occurrence as the prototype and all other occurrences have a flat structure, there might be a subset of prototypes, and different branches of how the prototypes repeat and evolve throughout the piece. Pattrans does not currently take this possibility into account.

Complexity in approximation and other transformations

In addition to the computational complexity that we discussed, the codes that implement the approximation transformations are rather complex conceptually. There are hardly easy ways to communicate to the machine what we see as approximations—the local deformations such as the insertions and deletions of notes, which are actually fraught with possibilities and complexities, despite their simple appearance at first sight. We do not claim that our implementation of approximation is optimal, and future research is needed to untangle this issue.

Other transformations, however, are selected with intuitiveness in mind, in the hope of reducing complexity. As we discussed in Chapter 4 and 5, there is a need for framing information coming out of computational methods in a more self-explanatory and understandable way to users. Meaningful interactions with the computations hinge on exchanging communicable results and making users less prone to misinterpretation. Our DSL, by employing modular transformations and operators to compose these transformations, is created by placing users at the centre of the design, which can be improved with multiple future iterations.

Transformations that were not considered

There exist a variety of other transformations in addition to the ones Pattrans. As we have explained in this chapter and motivated in the previous chapter, our criteria for implementing the primary and misc. transformations include how intuitive and how common they are. As an initial step that starts to look into the connections between patterns and transformations, we do not aim to be comprehensive. Instead, one of the advantages of Pattrans is that it is easy to add new transformations and combine them with existing ones. When used in combination with a variety of corpora, the list of the transformations will undoubtedly expand and even be learned from data, which is beyond the reach of this dissertation but still relevant for future study.

Chapter 7 Using Transformations to Understand Patterns

Music is the pleasure the human mind experiences from counting without being aware that it is counting.

– Gottfried Wilhelm Leibniz

Structuralism is the belief that phenomena of human life are not intelligible except through their interrelations. These relations constitute a structure, and behind local variations in the surface phenomena there are constant laws of abstract culture.

– Simon Blackburn

In this chapter, we will look at three applications and analysis of transformations using Pattrans with various musical corpora. We start with a recap of background knowledge we covered hitherto and an overview of the tasks we explore in this chapter. As a straightforward application, we first show how we can use transformations to query desired patterns using transformations and their combinations. From there, in order to study patterns from both algorithmic output and human annotation, we explore a variety of musical corpora and investigate how pattern occurrences are related and can be classified using transformations. Finally, we summarise our contributions in this chapter, and discuss the potential of our transformation-based approach for addressing crucial challenges of the pattern discovery task, such as the evaluation of pattern discovery algorithms, the analysis of human annotations of patterns, and the use of automatic pattern discovery for future corpora analyses on repetition and variation.

7.1 Overview

In previous chapters, we have seen that musical patterns are important for a multitude of reasons, and that algorithms have been devised to discover patterns using computational methods. Such musical patterns are widely discussed in different

contexts, such as the concrete MIR tasks of compression (Meredith, 2013), classification (Lin *et al.*, 2004), and segmentation (Nieto & Farbood, 2014), or the broader contexts of psychology (Foubert *et al.*, 2017), musicology (Janssen, 2018), and education (Harkleroad, 2006).

With such diverse applications, we see various definitions, examples, and criteria related to musical patterns. Due to this variability, it is non-trivial to design and evaluate an automated computational system to extract musical patterns in a way that suits every use case. Additionally, the automatic discovery of patterns remains a challenging task due to many other factors; for example, patterns discovered by different algorithms for the same piece tend to differ greatly, are often difficult to relate back to meaningful musical concepts, sometimes do not agree well with human annotations, and so on.

Inspired by functional programming, we made the connection between patterns and transformations. In a nutshell, with this connection, we emphasise *relations* between pattern occurrences: the repetitions and variations, within which we find all kinds of transformations. We implemented a variety of transformation checkers in Haskell for further comparing musical patterns. In this chapter, in accordance with the thesis statement of this dissertation:

Patterns, as a type of highly subjective and ubiquitous abstraction, have important connections to transformations, which can be explored compositionally using a functional programming language to reveal implications for designing musical pattern discovery algorithms.

We will use the transformations in Pattrans to relate musical pattern occurrences. Transformations are used as the criteria and explanation for determining whether an occurrence constitutes a pattern, in accordance with our working definition of patterns proposed in Chapter 2. More specifically, using the Pattrans package introduced in Chapter 6, we use a set of computationally well-defined and commonly used musical transformations to query and analyse pattern occurrences. As a result, we can retrieve and discriminate between different types of pattern, and subsequently use the proportion of each transformation to compare different corpora, pattern discovery algorithms, and annotation processes in this chapter. In addition, we will show potential additions we can make to the Pattrans package.

We aim to answer questions such as:

- Can we explore and retrieve patterns using transformations? (Section 7.2)
- Are the primary and misc. transformations able to capture the occurrence relations in human annotations? Are these relations similar or different in algo-

rithmically extracted patterns? How do the relations differ between corpora? (Section 7.3)

7.2 Querying for patterns

Pattern discovery is often divided into two different subproblems: knowing what we are looking for and investigate how to find it; and not knowing what we are looking for and investigate how to find it somehow (see more discussion back in Section 4.4) The former can be referred to as the occurrence or query problem and the latter, the establishment or discovery problem. Pattern discovery is sometimes confined to the stricter meaning of the latter. In this dissertation, we have taken a broader view on pattern discovery and therefore include both subproblems.

Querying for patterns is by no means a trivial or uninteresting problem. At least, the expressivity, efficiency, and testability considerations of this problem make it suitable for solving computationally and intellectually worthwhile to consider. First, expressivity can be a challenge for designing a query language: there is a trade-off between how expressive and specific the language can be and the size of the language vocabulary. Making a query computationally also requires a mental model that is aligned to the human who will be expressing the query, preferably in an intuitive way. Second, given a large amount of data and a complicated enough query, the execution efficiency of the query could be a bottleneck for retrieving the desired patterns. Third, evaluation-wise, once the specification of the query is expressed in the query language, the correctness of the query mechanism can be tested using unit or property testing methods. We, therefore, dedicate this section to demonstrating a method for querying for patterns using our transformations.

In effect, using the transformations and their checkers in the Pattrans package, we can get a pattern query language for free. Given a prototype pattern and a pattern checker (i.e. HomCheck Pattern, see Chapter 6 for more explanation), we can query the repetition of this prototype pattern by keeping a sliding window having the same size as the prototype and check the query against all such occurrences. The implementation of this can be seen in Code Snippet 7.1.

In terms of the complexity of this querying algorithm, note that all transformations mentioned in Pattrans in the previous chapter run in quadratic time, since they consist of linear pre-processing steps that transform the musical data in some way, followed by the quadratic approximation step. In the case of querying, we always check patterns of equal length N , so each individual check runs in $O(N^2)$. Moreover, we would need to perform $O(M)$ such checks in the average case via the sliding window

```

type W indowSize = Int
type Query a = (HomCheck a, a)

slide :: WindowSize -> [a] -> [[a]]
slide n xs = [ take n (drop m xs) | m <- [0..(length xs - n `max` 
  -> 0)] ]

query :: Query Pattern -> MusicPiece -> [Pattern]
query (checker, base) = filter (\p -> (base <=> p) checker) .
  -> slide (length base)

```

Code Snippet 7.1: Implementation of a simple query function.

method. Hence, the worst-case time complexity is $O(M^3)$. Luckily, the expected input size of a query will normally be a simple musical pattern of small constant size. Consequently, we would need to perform $O(M)$ checks, but each one would run on constant $O(1)$ time, resulting in an overall time complexity of $O(M)$.

Using the transformation, one can also specify transformations with respect to a certain prototype pattern and query those patterns. Then, the sliding window extracts all occurrences that satisfy the given checker specified by the transformations. We also provide a user-friendly interface for this in Pattrans, as shown in Code Snippet 7.2.

```

data UserQuery a = ToPattern a => Check Pattern :@ a

class ToPattern a where
  toPattern :: a -> MusicPiece -> Pattern

(??) :: ToPattern a => Song -> UserQuery a -> IO ()

```

Code Snippet 7.2: Define a query type for the user.

Notice that `UserQuery` is polymorphic over any type that is an instance of the `ToPattern` typeclass. One example of such a type is a pair of time points, indicating a start and an end time points in the song, as illustrated in the example typeclass in Code Snippet 7.3:

```

instance ToPattern (Time, Time) where
  toPattern (startT, endT) =
    takeWhile ((<= endT) . ontime)
  . dropWhile ((< startT) . ontime)

```

Code Snippet 7.3: A type class for `ToPattern` enabling using starting and ending time of a passage of music as the prototype pattern for the query.

A more flexible alternative is to provide a series of notes, in which case we use the Euterpea Haskell library, which provides a concise musical DSL (Hudak & Quick, 2018). In fact, we piggyback on Euterpea's MIDI export functionality to produce the

extracted patterns after discovery as well. We do not include all the source code to implement this in this chapter, and the rest of them can be found in Appendix D.

An example query based on time interval could be

```
"bach" ?? (transpositionOf ~~ 0.6) :@ (21,28)
```

which queries the transposition up to 60% approximation given the prototype pattern from time step 21 to 28 in a piece of music. Some example query results are shown in Figure 7.1 (a) - (b1-3). We can see that there are non-trivial variations of the base prototype pattern retrieved.

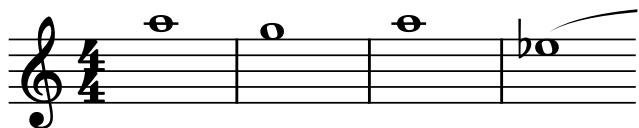
In the same figure, we also show two bars of music in (c), which consists of patterns retrieved by algorithms, where the second bar was retrieved as an occurrence of the first bar. In this case, our transformations are not expressive enough to relate the two occurrences, while they certainly bear similarity to each other. If one plays out these notes and listens, one might think of the first bar as a "question" and the second as an "answer". Although one cannot express this type of analogy with primary transformations, the primary transformations provide a clear boundary between the computationally easy-to-describe patterns and those that are not yet easily describable. The algorithm that originally retrieved the two bars in (c) might have solved the problem of how to describe this complex and subtle type of transformation, but, as this algorithm does not give a reason why these two occurrences constitute a pattern, we cannot be sure whether it is really the case that it has solved the problem by simulating a meaningful pattern discovery process or through other means.

With all these lessons we can learn from querying patterns, we urge designers of pattern discovery algorithms to also consider more concretely which patterns they are looking for in the first place, and develop on the query part of the pattern finding task. Furthermore, examining and interpreting several actual output patterns are also crucial steps in developing this process that should not be ignored.

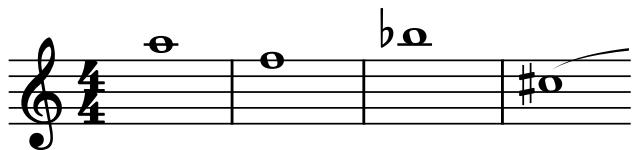
7.3 Analysing transformation data

In this section, we aim to demonstrate the transformations in Pattrans in a different capacity. We have seen that algorithms were devised to discover patterns using mathematical principles and music theories. However, it is not always straightforward to understand the output from algorithms and relate them back to meaningful concepts in music. The same is true for human-annotated patterns: due to the subjectivity of the annotators and the ambiguity in some musical material, human annotators often disagree with each other. To compare and interpret these patterns

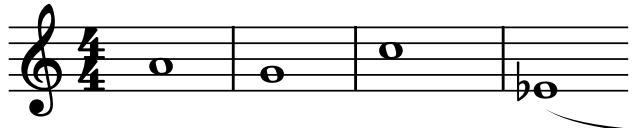
(a) Base for the query



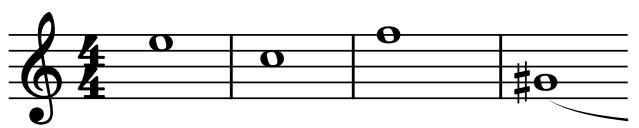
(b1) Retrieved occurrence with a similar interval structure



(b2) Retrieved occurrence that has been transposed to a lower octave



(b3) Retrieved occurrence that has a similar interval structure and has been transposed



(c) Algorithmically discovered pattern occurrences which cannot be characterised by primary transformations



Figure 7.1: Query examples. (a) is a query and (b1-3) are the queried results using a combination of transformations. We can see several intriguing intervals in these results. (c) shows an example of what cannot be queried and explained by the transformations we use.

through a musically relevant and objective lens, we use transformations to analyse the repetitions and variations between pattern occurrences. By relating pattern occurrences using transformations, we discriminate different types of patterns, and subsequently use the proportion of each transformation to compare different corpora, pattern discovery algorithms, and annotation processes.

We first introduce the data and algorithms we use. Then, we cross-compare human annotations from different corpora, which shows that a large proportion of the pattern occurrence relations can be explained by the transformations we consider, thereby establishing the importance of such transformations in human-annotated patterns. Next, we use six pattern discovery algorithms and show that the human annotations tend to contain higher percentages of exact repetition while the algorithmic ones tend to have higher percentages of misc. transformations (see Chapter 6 for the definition of misc. transformations). Following this, we use automatic pattern discovery and transformations to investigate the role of repetition and variation in different kinds of music. With these steps, we demonstrate the power of

transformations in modelling musical patterns in three interrelated challenges: the diversity of human-annotated musical patterns, the evaluation of pattern discovery algorithms, and the cross-corpora comparison of musical patterns.

We highlight the primary and misc. transformations introduced in Chapter 6. Primary transformations are exact repetition, chromatic transposition, and rhythmic-only repetition. Additional transformations, the compositions of transformations, and approximations, are grouped into "misc. transformations". All other occurrences which we could not match with any transformations nor approximations are grouped into "unclassified". The transformations are checked in this order as we described them in this paragraph (see more discussion about the order of checking transformation in Section 6.2 and 6.5).

After these three comparisons, we will make use of statistical analysis to explore how significant our comparisons are. We will also use the transformations to create new features and visualise them using PCA for visual examination.

7.3.1 Data and background

We begin by introducing more in detail the three sub-problems we will be tackling, as well as the dataset we will be using. As there are overlaps of datasets across chapters, we do not reiterate all the details here, but summarise each dataset in more detail in Appendix A.

Human annotations

We have seen that datasets of human-annotated musical patterns are rare, and agreement amongst human annotators is rarer still. These disagreements can be attributed to ambiguity in the musical material, subjectivity of the annotators, different annotation protocols, and different annotation tools used. As a result, controversies remain when it comes to using human annotations for musical pattern discovery research, and other MIR tasks. In order to study the differences and commonalities between different annotators, we can now employ primary and misc. transformations to compare different pattern occurrences by grouping these according to their relations, which sheds light on the potential criteria the annotators have used. Together with the [JKU-PDD](#) and [MTC-ANN](#) dataset, we use the dataset collected by ANOMIC for this purpose. We will call this dataset the **HEMAN** dataset in this section because the musical pieces used are identical to the initial HEMAN experiment. We separate this dataset into two sub-datasets according to the annotator's musical backgrounds: HEMAN-High (with a high

self-rated musical background) and HEMAN-Low (with a low self-rated musical background).

Algorithms

We have seen that while datasets of human annotated patterns are scarce, there exists a wealth of patterns produced by musical pattern discovery algorithms. Persistent challenges with evaluating and deploying such algorithms exist: there can be a large number of output patterns, which are costly to examine manually; implementation logic can be hard to comprehend, which could be caused by any of the numerous procedures that comprise the algorithm, or by a binary-only release for which we only have access to the output; patterns extracted by different algorithms can vary hugely given the same input, which creates controversy when it comes to using human annotations as ground truth and designing an all-encompassing evaluation strategy. The above usability, intelligibility, and evaluation problems can also be seen in many other domains of MIR research, and more generally, in algorithm design and software engineering. The transformations provide us with a lens through which to examine the output using musically relevant concepts, therefore enabling straightforward understanding and comparison of the algorithms.

For investigating patterns from the algorithms, we use the six pattern discovery algorithms submitted to the [MIREX](#) task during 2014-2017. Each algorithm has its own series of procedures to extract patterns from music, e.g. clustering, matching, and counting, and so forth. The number of patterns extracted varies across algorithms and corpora. We run the following algorithms¹ on the JKU dataset (the number of patterns extracted are in parenthesis): SYMCHM (59), SIACF (794), SIACR (1590), SIACP (399), VM1 (2969), VM2 (463). We use the following corpora with the SIACF algorithm (the number of patterns extracted are in parenthesis): Euro-Vision (7696), Jazz (6016), HEMAN (322), MTCANN (4486). Each pattern has at least two occurrences as defined in MIREX.

Corpora

We have seen that, across a wide range of corpora in various styles, transformations have been used and analysed in the context of music theory and musicology. With our computational approach to transformations, we envision facilitating corpora comparisons regarding these generic transformations in order to answer musical questions on repetition and variation. For example, as we saw in Section

¹we introduced these algorithms in Chapter 4

5.3, it was proposed in (Margulis, 2014) that the amount of repetition varies between compositions created by different composers in different styles, and it was proposed in (Middleton, 1990) that popular music is highly repetitive. These musicological enquiries about repetition and variation can be supported by our approach: if the above arguments are true, we should be able to find supporting evidence in the patterns and their occurrence relations using transformations.

7.3.2 Transformations in human annotations

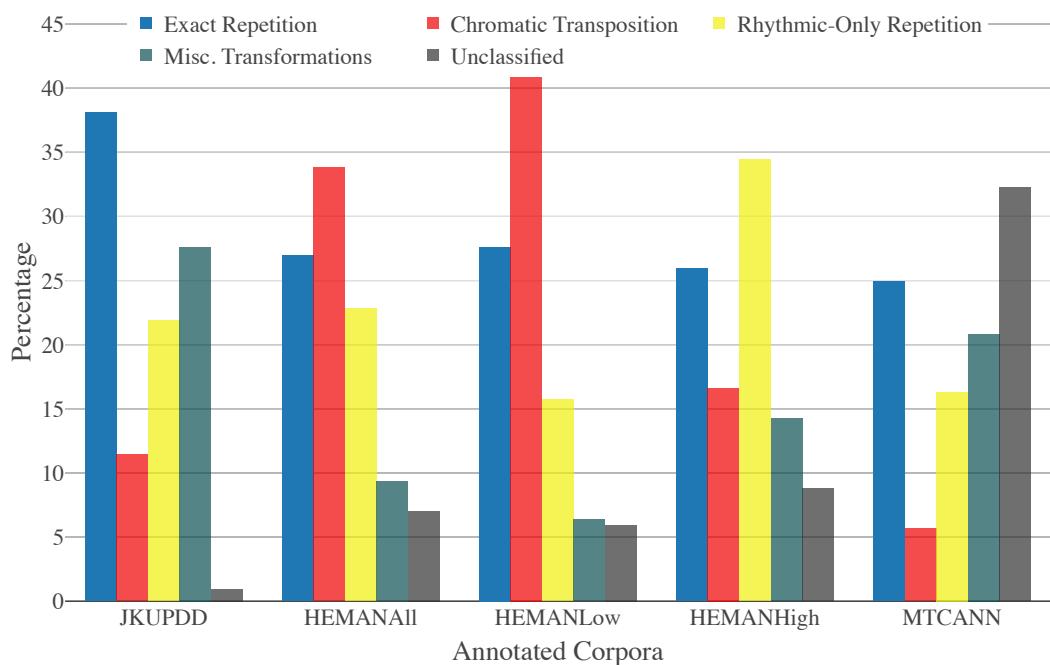


Figure 7.2: Transformations found in the human-annotated patterns in 3 datasets and 2 sub-datasets. The colours correspond to different transformations.

In Figure 7.2, across three datasets and two sub-datasets, we consistently see a substantial proportion of exact repetition. Exact repetition, chromatic transposition, and rhythmic repetition together make up the majority (71.4% for JKUPDD, 83.6% for HEMAN, 87.7% for HEMAN-Low, 77.0% for HEMAN-High) of the pattern occurrence relations, with exception of MTCANN (46.9%)—although it does have a percentage close to 50%. This is supporting evidence that exact repetition, chromatic transposition, and rhythmic-only repetition are of major importance to human-annotated patterns. Additionally, as the proportion of unclassified relations (occurrences that could not be related with the transformations in Pattrans) tends to be small, and the proportions of those relations classifiable by transformations are above 70% for all datasets, this is a positively validating result for the use of our transformation approach.

The MTCANN dataset has the most unclassified occurrence relations. In Figure 6.2, we have already seen that pattern occurrences in this dataset can be short but diverse. The relations between the occurrences in Pattern 1 will be identified as "unclassified" because 50% (the first two bars) are not matched. In Pattern 2, because we only have two notes in these occurrences, with a one-note difference, we can also only identify these occurrences as "unclassified". In general, a minor change in a short pattern can render the occurrence relation unclassifiable. It is an indication that recovering the human-annotated patterns using an algorithm in this dataset is difficult.

The annotations in the HEMAN dataset, as described in Chapter 3, are gathered using a tool with automatic exact repetition and chromatic transposition tagging. A potential side-effect is that we see a higher proportion of chromatic transpositions in HEMAN and HEMAN-Low than other datasets. This raises awareness of the possible side effects caused by providing the functionality of automatic repetition tagging in annotation tools. Annotators with higher self-rated musical background scores, in HEMAN-High, had a higher proportion of rhythmic repetition, which is an indication that these types of rhythmic structures are more prominent as musical patterns for the annotators with more musical background. We will also see in Section 7.3.4 that there are more rhythmic repetition in the algorithmically extracted patterns in the HEMAN dataset than in JKUPDD and MTCANN.

Looking across the datasets, despite certain high-level consistencies in the proportions of exact transformations, we see various proportions of misc. transformation. This could be the result of the different annotation gathering processes: with different applications and goals when annotating the patterns, together with different levels of expertise, and hypothetically different schools of thoughts and theories, human-annotated musical patterns can be diverse. As we will see in Section 7.3.3 and 7.3.4, the results from different algorithms given one fixed corpus show diversity too, while patterns extracted by one algorithm from different corpora are more consistent.

7.3.3 Transformations in algorithmic output

Comparing the output of six algorithms to the human annotations of JKUPDD, in Figure 7.3, we observe different combinations of proportions of transformations. More specifically, we make the observation that although most of the algorithms retrieve smaller proportions of exact repetition than in the patterns annotated by humans, all algorithms extract a nonzero number of exact repetition. SYMCHM is the only algorithm with more exact repetition and chromatic transposition in the

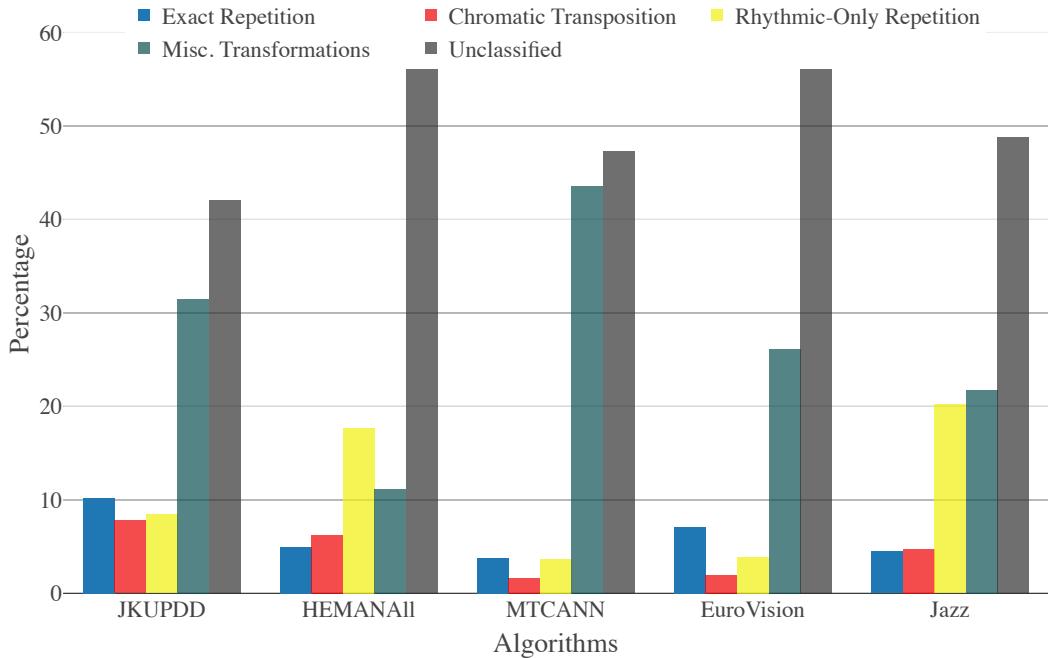


Figure 7.3: Transformations found in human-annotated and algorithmically extracted patterns in the JKU dataset. Using the same specifications as Figure 7.2.

proportion of the extracted pattern occurrence relations. It does not, however, extract patterns with rhythmic repetition. VM1 and VM2, belonging to a family of wavelet-based pattern discovery algorithms, have the most unclassified pattern relations. SIACP and SIACR have more misc. transformations and fewer chromatic transpositions than other algorithms.

Correspondingly, we connect these observations with more background on the algorithm. In the JKUPDD dataset, human-annotated pattern occurrences are mostly related to each other by straightforward exact repetition and chromatic transposition. Except for SYMCHM, other algorithms extract pattern occurrences with less exact repetition in proportion. The SYMCHM algorithm employs a machine learning approach. Transformations are not explicitly encoded in the algorithm. Nevertheless, we observe that a large proportion of the output can be classified into different transformations, which indicates that the algorithm is able to learn these transformations. In this way, we interpret the output of the algorithm using domain-specific concepts, which have valuable potential to phenomenologically verify and multi-facetedly interpret a complicated algorithm. The particularity that this algorithm does not extract rhythmic-only occurrences coincides with the observations in (Ren, Volk, *et al.*, 2018) that the rhythmic features of the pattern occurrences are largely different from other algorithms and human annotations.

In VM1 and VM2, a clustering algorithm was used as the final step in grouping the pattern occurrences. The large number of unclassified relations could be due to this clustering step, whereby the distance metric used in the clustering algorithm could have captured relations that cannot be matched to the transformations we considered.

SIACP, SIACR, and SIACF belong to the geometric family of pattern discovery algorithms. The names of the algorithms correspond to different parameter settings concerning the maximisation of the recall (SIACR), precision (SIACP), and F-1 score (SIACF). We see that patterns discovered by the three algorithms have occurrence relations mostly in the category of misc. transformation, where there is a diverse combination of transformations and approximations. Notably, chromatic transpositions are not present in SIACP and barely present in SIACR. This could be a corpus-specific phenomenon or an artefact of the combined effects of the filtering steps in the algorithms.

These observations on the differences between the algorithms also give us more grounds to discuss which algorithms we should choose for different application scenarios, and how to evaluate their performance thereafter. For example, if the pattern discovery process were to be designed for reproducing human annotations in the datasets we examined, the SYMCHM algorithm could be used with the awareness that rhythmic-only repetitions need to be incorporated separately. In addition, we should also consider that, depending on context and purpose, annotations are gathered using different strategies and protocols, as we have seen in Chapter 4. For example, annotated patterns from a musicologist to analyse the structural elements of music are likely to be different from the patterns found by a music therapist for informing their suggestions (Foubert *et al.*, 2017) or music teachers for pointing out pedagogically interesting passages. In these specific cases, having a perspective on what the relations are between the annotations can help with choosing one or a combination of algorithms. If the goal is to reveal hidden patterns and relations for the task of compression or for a deeper mathematical understanding of the music, wavelet and geometric-based algorithms are suitable for the task. If the application's goal is to retrieve patterns for composing new music (Herremans & Chew, 2017), the choice of the algorithms can be based upon the envisioned relations between different parts of the music.

7.3.4 Comparing patterns across different corpora

In addition to the JKUPDD, MTCANN and HEMAN we examined, in this section, we now go on to examine two more datasets, EuroVision and Jazz, using the SIACF

algorithm. We choose this algorithm to examine the datasets partially because its occurrence relations, as shown in Figure 7.3, do not have a large proportion of "unclassified" relations (unlike VM1 and VM2) nor misc. transformations (unlike SIACP and SIACR), and partially because it is readily available and open-source (unlike SYMCHM).

Figure 7.4 shows the different proportions of transformations found in different corpora using the SIACF algorithm. The performance of a single algorithm across datasets can provide us with an objective measure of the pattern occurrences in the corpora. An alternative could be to use sliding windows to compare all possible segments in music with one another. However, the exponential growth of the possibilities renders this computationally intractable. With the help of pattern discovery algorithms, we significantly reduce the relations we need to check. Looking across all datasets, we see a large proportion of "unclassified" and misc. relations, which shows that the patterns retrieved by SIACF are largely non-trivial, as we have shown in the previous two sub-tasks.

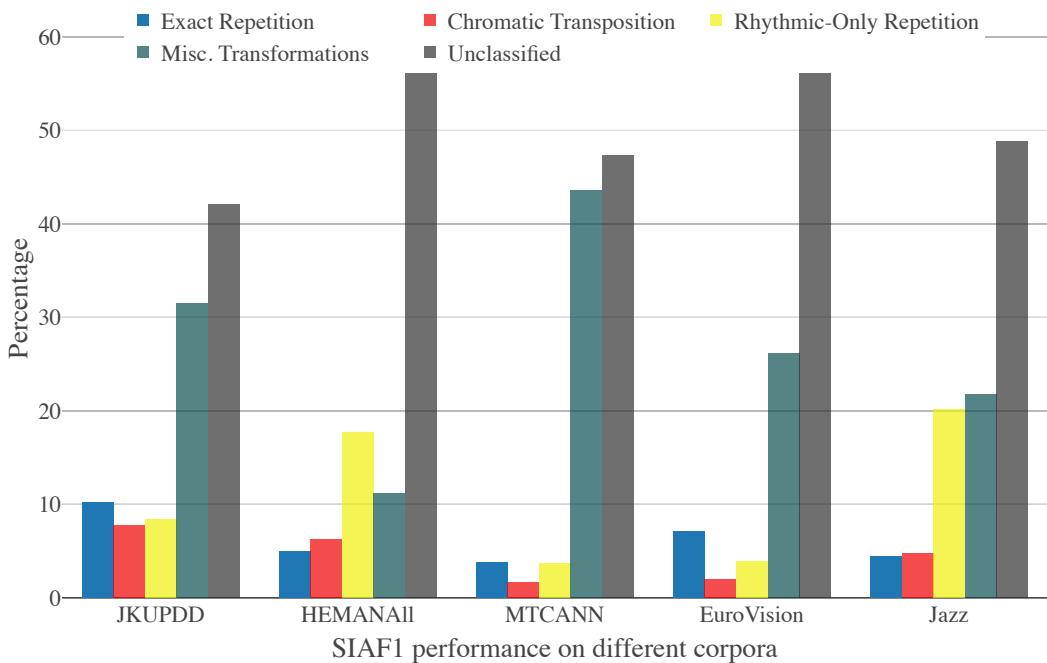


Figure 7.4: Transformations found in the output of SIACF using five different corpora. The values on the y-axis represent the percentage of the transformations found in the discovered pattern occurrence relations.

More specific observations on each dataset are as follows. JKUPDD, which contains pieces mainly in the classical style, has the largest proportion of exact repetition when applying SIACF. This is followed by EuroVision, which contains music mostly in the pop style, and HEMAN, which is mostly classical as well. The least propor-

tions of exact repetition are found in MTCANN (folk) and the jazz solos. Regarding chromatic transposition and rhythmic repetition, HEMAN and jazz have the largest proportions, followed by JKUPDD. In MTCANN and EuroVision, we see smaller proportions of chromatic transposition. MTCANN contains the most misc. transformations in percentage, followed by JKUPDD. The remaining three datasets have less than 30% misc. transformations.

Through comparing the human annotations in different corpora in Figure 7.2 and 7.4, we see that the annotators with higher self-evaluated music background score agree more with the SIACF algorithm in that they both have a higher proportion of rhythmic-only pattern occurrences in HEMAN than the JKUPDD and the MTCANN dataset. Although highly speculative, with future work that covers a wider range of algorithms and corpora, this could constitute supporting evidence that the algorithm can be used to help with pattern discovery.

As we only use one algorithm to demonstrate this type of investigation, we cannot assume a high level of generalisability of these results. Nevertheless, we do show how to use a pattern discovery algorithm to compare the number and types of repetitions and variations in corpora of different styles.

7.3.5 Statistical analysis

In the previous three sections, we compared transformations in patterns with different datasets, annotators, and algorithms. To further investigate these differences, we leverage statistical methods to test their significance.

There has been criticism levelled against significance tests. As advocated in (Scheel *et al.*, 2020), there are more important processes such as the *derivation chain*: "a conjunction of theoretical and auxiliary premises that are necessary to predict observable outcomes". We agree that more effort should be spent on forming concepts, developing valid measures, establishing the causal relationships between concepts and their functional form, and identifying boundary conditions and auxiliary assumptions, as we did in previous chapters. We employ statistical testing here simply as an exploratory method for the data we produce from Pattrans.

To further confirm the significance of the group difference between human annotations and algorithmically extracted patterns, we performed a statistical test, the Kruskal-Wallis one-way analysis of variance, with the null hypothesis that there is no significant group difference between the algorithmically extracted patterns and the annotated patterns, and with the significance level $\alpha = 0.05$.

The Kruskal-Wallis test is a non-parametric method for testing whether samples, i.e. the proportions of transformations, originate from the same distribution. It does not assume a normal distribution of the residuals in the samples.

For the MTCANN dataset, we obtain $p - \text{values} \ll 0.05$ for both exact repetitions and transpositions. The null hypothesis is rejected. The effect size, as we computed, however, is small with the value of 0.1873545. Therefore, the group difference of the exact repetitions and transpositions between the annotations and the algorithmic output is significant but small.

For the JKUPDD dataset, we obtain $p - \text{value} \ll 0.05$ for exact repetitions, whereas $p = 0.109 > 0.05$ for transpositions. The hypothesis is rejected for the proportion of exact repetitions, suggesting a significant difference between the two groups. We cannot state the same for the proportion of transpositions in the JKUPDD dataset: the proportion of chromatic transpositions in human and algorithmic patterns seems to be not significantly different.

To show the statistical aspects more concretely, in Figure 7.5, we show a specific transformation, the exact repetition, by plotting a boxplot with Kruskal-Wallis performed three folds (see three p-values on the upper part of the figure) to compare the algorithmic patterns and human-annotations. We can see that SYMCHM and VM1 have a non-significant p-value while SIACF1's p-value is significant. This exposes the individual differences between the algorithms. More specifically, the output from SIACF1 has a significantly different amount of exact repetition as in human annotations, and differences from the other two algorithms are not significant when $\alpha = 0.05$.

The differing proportions of transformations could inform the future design of pattern discovery algorithms. If the authors of these algorithms wish to imitate the pattern discovery behaviours of human annotators, they should report simpler transformations in pattern occurrences that have musicological and music theoretic support. The purpose of the algorithms, however, is not confined to the replication of the reference data. More generally, calculating and comparing the proportions of transformations can serve as an extra step to answer questions about algorithms, such as: does the algorithm discover simple patterns in a way that can be explained via the musical concepts of transformation? Or is the algorithm trying to be more "innovative" in the patterns they discover?

Given the importance of these transformations, in the next section, we propose a set of new features based on the proportions of transformations, which we call the *transformation profile*. We then use PCA on these TPrs to further compare ex-

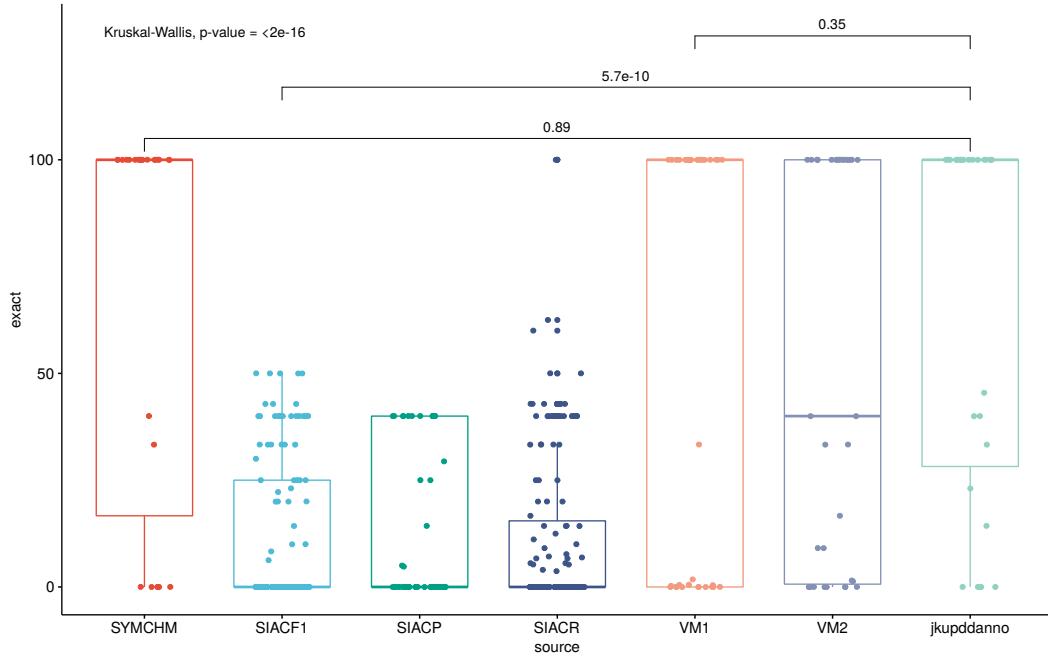


Figure 7.5: Boxplot and Kruskal-Wallis test of exact repetition.

actly how the human-annotated and algorithmically extracted patterns differ in this transformation feature space.

7.3.6 A transformation profile for each pattern and PCA

As transformations characterise the relations of occurrences in each occurrence set, we can use the proportion of each transformation to describe patterns. More specifically, we can create a feature vector based on how often a transformation appears in the occurrence set.

For each pattern, we define a **transformation profile** from the proportion of different transformations:

$$TPr(\text{Pattern}) = \left\{ \frac{c(t_1)}{\sum_{i=1}^n c(t_i)}, \frac{c(t_2)}{\sum_{i=1}^n c(t_i)}, \dots \right\} \quad (7.1)$$

where t_i is a certain transformation indexed by i , $c(t_i)$ is the count of a certain transformation, and n is the number of different transformations. The denominator is essentially the count of all occurrences if we take into account the "unclassified" as a type of unknown transformation.

Intuitively, the feature vector is the numeric vector representation of the bar charts presented in Section 7.3.2, 7.3.3, and 7.3.4, that is, the constituents of occurrence relations. These are, in fact, the TPrs as defined in Eq. (7.1). By representing the

patterns using TPr, we can further investigate and compare the patterns in this feature space spanned by the proportion of exact repetition, transposition, and other transformations.

Each transformation corresponds to a dimension in our TPr. The more transformations in our analysis, the greater the number of dimensions the TPrs will have. To examine a higher-dimensional TPr, we can use multidimensional scaling [MDS](#) methods to visualise it. In this visual way, we can show how patterns are located in terms of the transformations they contain. Furthermore, we can compare precisely how the human-annotated and algorithmically extracted patterns differ in this transformation feature space.

In Figure 7.6, we plot each individual pattern as a data point in the PCA decomposition of the TPr feature space. For each pattern, we colour and size them differently depending on their source: algorithmically extracted or human-annotated.

The PCA is calculated based on the JKUPDD annotations, and the algorithmic patterns are projected into the space for comparison. From the calculation, the first Principal Component (PC) explains 63.6% of the variance, and the second PC explains 23.6%. The fact that the main contributions to the first and second PCs come from exact repetitions and transposition highlights the importance of simple transformations and approximation. Moreover, by examining the principal components and the graphical depictions provided by the PCA projection, we are able to obtain a better view of the collective traits of the transformations in musical patterns.

To interpret the figure, we first notice that data points form a linear relationship in this feature space. It follows that there are correlations between the axes. It is expected that the data will exhibit this correlation since the proportions of transformations sum to 1 and are therefore not independent from each other.

We can further make some key observations that provide more validity to the algorithmic output. What is immediately obvious to us is that the algorithmically discovered patterns, projected into the PCA space, tend to interpolate the anchoring human annotations. In simpler terms, the algorithmic output fills in the space between the annotations.

Despite the different proportions of transformations reported in previous sections, the algorithmic data points exhibit directional similarity to those of human annotations in the PCA space. The direction (spread) is a visualisation of the variance in the data. Sharing similar spreads suggests similar variations in these data points of TPrs. It provides support that the algorithmic output conforms to the variance of the proportions of transformations in human annotations. This provides a certain degree of validity for the algorithms.

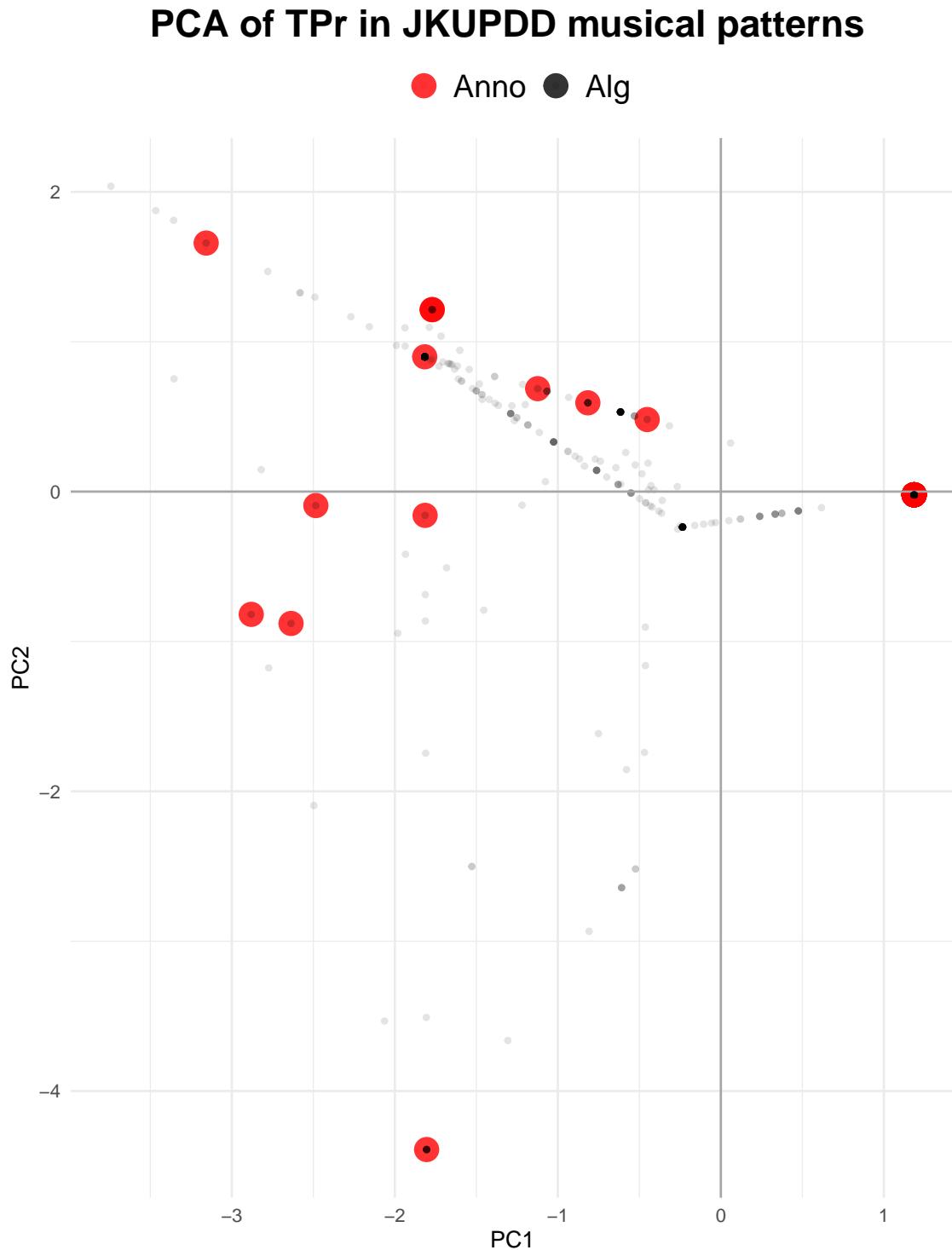


Figure 7.6: PCA visualisation of patterns' TPrs of human-annotated and algorithmic patterns in JKUPDD. The highly ranked transformations in the first Principal Component (PC1) are exact repetition (37.24%), misc. transformation (29.42%), and rhythmic repetition (22.45%) and for the second Principal Component (PC2), transposition (76.10%), and misc. transformation (15.19%).

Admittedly, gathering more annotations could falsify this specific observation. However, this also motivates future work involving collecting more human-annotated musical patterns to further the research into algorithmic musical pattern discovery.

In addition to the 2D visualisation, we provide 3D interactive visualisations using multiple MDS methods such as *t-distributed Stochastic Neighbor Embedding (tSNE)* and *kernel Principal Component Analysis (kPCA)*². Figure 7.7 is a screenshot of the 3D PCA visualisation interface. For each individual pattern, we colour and mark them differently depending on their sources: algorithms or musical pieces. The additional third dimension allows us to have more direct access to the spread of data in the TPr feature space. By interactively engaging with the visualisation, we can examine the TPrs of individual patterns and musical pieces as we please.

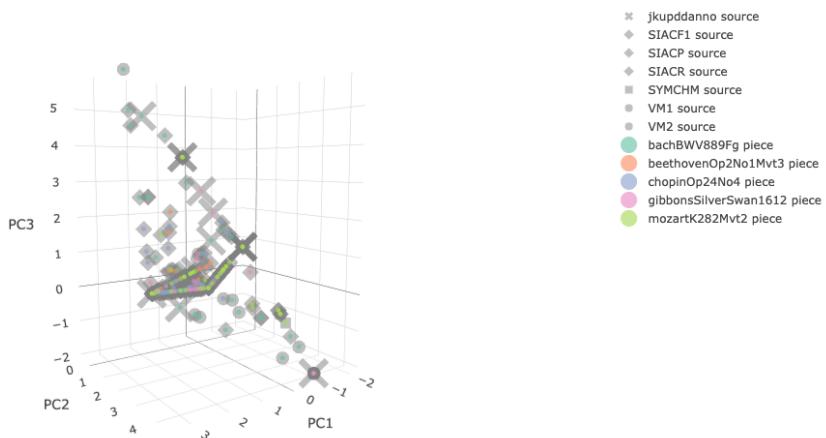


Figure 7.7: Interactive 3D PCA visualisation of the patterns' TPr. In addition to the two dimensions in Figure 7.6, the third dimension mainly consists of rhythmic repetition (69.61%) and misc. transformation (21.03%),

7.4 Summary

In this chapter, using transformations, we captured the notion of repetition and variation in more detail. We developed a computationally and conceptually tractable framework for analysing musical patterns and pattern discovery algorithms.

First, we presented the query function of Pattrans that allows us to use transformations to query a given musical piece. Although it suffers from a naive cubic running

²Use this URL with different endings such as pca, tsne, and kPCA: <https://irisyupingren.github.io/research/3d/pca>

time complexity, it proves useful in practice with queries of small size. Since the DSL is embedded in Haskell, we can re-use existing music libraries in tandem with our framework, as exemplified in our use of Euterpea musical expressions as queries for pattern discovery.

Second, through the systematic lens of the transformations, we compared different human annotations, algorithmically extracted patterns, and corpora. The cross-comparison of human annotations from different corpora shows that a large proportion of the pattern occurrence relations can be explained by the transformations we consider, thereby establishing the importance of such transformations in human-annotated patterns. We used six pattern discovery algorithms and showed that human annotations tend to contain higher percentages of exact repetition and other primary transformations. In general, human annotations show less variety of transformations and approximations to cover the relations amongst pattern occurrences, while the algorithmic ones show more complex transformations. Nevertheless, we also see that both algorithmic and human annotation processes can produce complex patterns given certain datasets. We also used algorithmically extracted patterns to compare different types of transformations across five corpora. We obtained some quantitative evidence for music theoretical hypotheses about repetition and variations.

In summary, we present two take-home messages about primary and misc. transformations with regard to: i) human-annotations and algorithmically extracted patterns, ii) cross-corpora comparison with these human-annotated and computed patterns:

- Primary transformations account for a large proportion of human-annotated pattern occurrence relations; by contrast, they only account for a small proportion of algorithmically extracted pattern occurrence relations, which contain more misc. and unclassified transformations.
- Using a pattern discovery algorithm, we can objectively compare different types of transformations across corpora of music in various styles. We see indications that primary and misc. transformations can be used to characterise corpora with different types of repetitions.

In addition to these above-mentioned comparisons, we used statistical methods to analyse the significance of the differences. We also devised TPr to characterise the algorithmically extracted musical patterns and compared them with human annotations. With our PCA and classification results, we showed that, in terms of transformations, the algorithmically discovered pattern occurrences conform to human annotations to a certain degree.

7.5 Discussion

The importance of understanding repetition and variation can be seen in a variety of domains: art theory, cognitive theory, music theory, and more. One of our approaches showed that computational methods can be helpful in substantiating different schools of theory in these domains. Generally, when there is a disagreement about different types of repetition and variations, transformations can provide quantitative summaries, which provides a basis for extra insights. In this chapter, not only did we untangle the complexities of musical patterns by employing the meaningful musical concept of transformations, but the discussion can also be extended to closely related concepts such as musical similarity and musical variation. In the rest of this discussion section, we will discuss a few more implications and limitations of our method.

Regarding similarity

Through transformations, we can gain a different perspective regarding musical similarity because the complexities of the transformations can be indicative of different degrees of similarity (Hahn *et al.*, 2003). Drawing on Meyer's endorsement of the idea that it is easy to derive any musical idea from any other musical idea, a question we can ask ourselves is: is it *equally* easy to derive any musical idea from any other musical idea or is there a spectrum of difficulty when deriving one musical idea from another? Our results provide evidence for the latter. If we assume that deriving a pattern occurrence from another while preserving rhythm is "easier" than changing both pitch and rhythm at the same time, then the different proportions of rhythmic-only, misc., and unclassified transformations, as shown in Figure 7.2, indicate that there is a difference when deriving one pattern occurrence from another.

Regarding distance measures

Distance measures are of much importance in MIR research. As a dual to similarity we mentioned in Section 2.3.1 and 5.3.6, different distance measures have been used in MIR research: correlation distance, city block distance, Euclidean distance, Earth Mover's distance, and so on (Janssen *et al.*, 2017; Typke *et al.*, 2005).

There are interesting connections to be found between our transformation-based approach and the distance/similarity-based approach to studying musical patterns. Let d be a distance measure, and g be a transformation, and x be a musical pattern

occurrence. We can then measure the distance before the transformation and after, which gives us $d(x, g(x))$. Depending on which distance measure is used, and whether there is a transformation involved, we may be able to use them to compare difference aspects of repetition and variation to different degrees. For example, when $d(x, g(x)) = 0$, the distance is invariant under the transformation, or the transformation is an identity transformation— $g(x) = x$.

Advancing musical pattern discovery

In broad terms, our results have the following additional ramifications. We hope that they will grant us further opportunity to advance, first, the field of computational approaches to pattern discovery and characterising different corpora, and second, the phenomenological interpretation of algorithms.

We can use musical transformations to examine and compare the corpora, the algorithms, and the annotation processes. This allows us to answer questions such as whether there are more repetitions present in one corpus than another (Margulis, 2014; Middleton, 1990); whether some musical patterns contain more diverse occurrence relations and are thus more complex than others; and whether different human annotators and algorithms extract systematically different patterns and how we can compare and reference them to one another. For example, it has become clear that if we choose to use human annotations as the sole reference for evaluating algorithms, we will not be able to recognise the strengths of an algorithm that is better at finding transformations that might not be immediately recognisable by humans, yet be important for describing musical structures in other contexts.

More generally, given a complicated process, be it a human annotating patterns or an algorithm learning and extracting patterns, we can compare their results on a finer level by reorganising them into easy-to-interpret and domain specific "baskets" (primary and misc. transformations in our case of music). Similar lines of thought can be seen more generally in interpreting machine learning algorithms using rule-extraction and relevance analysis methods (Guidotti *et al.*, 2018; Macdonald *et al.*, 2019). Our approach can also be useful for interpreting the output of other black-box/poorly-documented/binary-only/complex implementations. By classifying algorithmic results into finer categories such as "the expected", "complex result", "misbehaviour", we may help algorithms and humans to be more perceptually aligned.

We did not provide a ranking of the algorithms

Readers might expect a final ranking of the performances of algorithms at the end of this chapter. We do not conclude with such a ranking for two reasons. Firstly, using transformations is only one way to look at which algorithms match most closely to the human annotations, and sometimes it is even desirable for the algorithms and human annotations to have different transformations: a case-by-case analysis is more informative than an overall ranking. Secondly, we do not attempt to provide utilitarian feedback on which algorithm or transformation is more useful. Utility is subjective. Instead, we argue that different algorithms should be used for different purposes and in different contexts: education, MIR, and composition, just to name a few. Instead of using a single metric such as *accuracy* or *cross-entropy*, we provide richer insights by applying more semantically meaningful considerations.

Admittedly, we could have used transformations to produce a numeric mapping between algorithms and different applications, such as a conclusion that algorithm A is better than algorithm B for this purpose because the transformations produced by A fit better in this context. However, we do not have enough data regarding which transformations are more desirable than others for which application. We will discuss some more limitations in the remainder of this chapter.

Other limitations

It is not trivial to find transformations that can characterise repetitions in corpora as well as capture how casual listeners perceive patterns in music, and we bypass this problem for the time being by using transformations that are generally widely used. We do not claim that the transformations are comprehensive and novel—we instead focus on using these constrained transformations in a computational context to provide insight into the differences between different annotators, algorithms, and corpora. We also do not touch upon the intentionality behind the musical variations, which is outside the scope of this work. We are also aware that there are almost certainly other annotations, algorithms, and corpora in the wide world that are not featured in this dissertation. We leave these for future work.

Chapter 8 Conclusions and Future Work

People know what they do; frequently they know why they do what they do; but what they don't know is what what they do does.

– Michel Foucault

World is the pattern of meaningful relations in which a person exists and in the design of which he or she participates.

– Rollo May

In this dissertation so far, we have extensively discussed one of the widespread particularities of music—patterns, as well as the connection to automating the discovery of patterns in music. Although we will not have the space to more deeply discuss other aspects such as the social functions and the emotional connection we have as humans to music and its patterns, in this last chapter of the dissertation, we summarise what we have learned in previous chapters. We discuss a few potential applications of what we have learned in particular. Following that, we look in two directions—back into this dissertation and into the future work. Finally, we will close this dissertation by discussing other features and important themes in musical pattern discovery.

8.1 Summary of the chapters

In Chapter 1, we started by laying out our motivations, expected challenges, disciplinary contexts, scopes, thesis statement, structure and conventions used in the dissertation. We hope that we have discussed all the issues we set out to do, as we shall see in summarising other chapters below.

In Chapter 2, we moved on to examine the concept of pattern from a range of perspectives. We introduced the importance of repetition and variation and made many connections with other related concepts. We arrived at a working definition

of pattern that emphasises the fact that an explanation should be given if one were to consider a musical passage as a pattern. In subsequent chapters, we see that this explanation could be the fact that an algorithm or a human annotator chose them, it could be an array of occurrences, or it could be the transformations between the pattern occurrences.

In Chapter 3, we dived into gathering human annotations, designed annotation tools, and analysed the gathered data using the positions and features of the annotated patterns. Our results showed that many factors should be taken into consideration when gathering and using human-annotated musical patterns, such as the annotators' musical backgrounds, the interface of the annotation tools, and the protocol of the annotation experiments. Several considerations on using human reference for evaluating algorithms were also discussed.

In Chapter 4, we examined a range of pattern discovery algorithms and proposed five methods to evaluate some of them, including using visualisation, PP, CC, synthetic data, and prediction. We learned that there are differences in many different dimensions when it comes to the musical patterns extracted by different algorithms. One of the most significant differences comes from rhythmic features.

In Chapter 5, we changed our direction and looked at a variety of music-related software and DSLs. Given the advantages that can be provided by functional programming and Haskell, we propose to make the connection between pattern and transformation. We examined a variety of transformation-related literature to support our idea, and identified two types of transformations for the implementation in the next chapter.

In Chapter 6, we listed and described the individual transformation we would be using, and implemented Pattrans, our own DSL for comparing musical patterns. We make use of Haskell's type system and ideas from category theory for an implementation where it would be easy to combine and modify the transformations we proposed.

In Chapter 7, we used Pattrans in three different scenarios for querying patterns, comparing patterns quantitatively. We showed that by using transformations, we could obtain novel insights and practical applications by quantitatively analysing repetition and aligning human concepts with algorithmic output.

8.2 Applications

In previous chapters, we have already seen some potential application of musical patterns in the domain of MIR tasks, music research and analysis, HCI, and edu-

cation. In this section, combining with what we have seen in this dissertation, we summarise and extend our discussion with four applications mentioned in previous chapters.

8.2.1 Data collection

This is an application we saw first in Chapter 3: the helper function in ANOMIC reduces the repetitive steps in a potentially labour-intensive annotation process. It may sound circular, but there is a loop between gathering data and designing algorithms. We can use algorithms to gather more data, and data helps to improve algorithms, with humans being in control of the quality of data.

8.2.2 Patterns for Education

In music education, (Collins *et al.*, n.d.) also mentions that "the patterns could be interesting to teachers of music theory who are looking for examples of repetition on small and large scales, to differing degrees of exactness, and from a variety of musical periods." In other education domains, (Geist *et al.*, 2012) shows that musical patterns facilitate understanding mathematical principles such as spatial properties, sequencing, counting, patterning, and one-to-one correspondence. Three types of patterns have been observed, including repeating patterns (repeating sequences), growing patterns (e.g. count by twos), and relationship patterns (e.g. multiplicative sequences). For programming, there are many systems that aim to teach computational thinking through music (Dannenberg *et al.*, 1984; Freeman *et al.*, 2014; Ruthmann *et al.*, 2010). Although there is a entirely different meaning in using the word "pattern" in computer science, there is a similarity between the patterns we discussed and those used in a software technology context: " (patterns) capture the essence of the practice in a compact form that can be easily communicated to those who need the knowledge" (Bergin *et al.*, 2012). With the right approach, musical patterns can be used as a vehicle that promotes computational and abstract thinking.

8.2.3 Creativity and algorithmic composition

Creation in the music domain can be both easy and difficult. A simple concatenation of patterns can yield interesting results; a few variations can help construct a whole piece. However, it is easy to have musical tropes and make the mistake of juxtaposing incompatible patterns and emotions. Pure imitation and copying of patterns can hardly be called creative, and there is a tension between creativity and control. The

8 Conclusions and Future Work

relations between and the order of patterns, however, can be more sophisticated and subsequently instrumental to creativity research. How can we use patterns to bottle human creativity in order to archive prolific machine creativity?

Let us take a step back and start over with the question: what is creativity? (Boden, 1996) defines creativity as "the ability to come up with ideas or artefacts that are new, surprising and valuable". Three types of creativity were proposed: transformational, exploratory, and combinational. (Beghetto & Kaufman, 2007) categorises creativity differently: professional, everyday, and personal. In another context, creativity has been defined as "the production of novel and useful ideas" (Amabile, 1988). It often involves a process of blind variation and selective retention (Campbell, 1960; Simonton, 2013), and computers are particularly good at blind variation.

Computational creativity and musical creativity are research topics on their own (Colton, Wiggins, *et al.*, 2012; Deliège & Wiggins, 2006; Wiggins, 2006). Algorithmic composition, a related to subject area, has also become a vast subject associated with a myriad of methods. More and more research is being conducted in the direction of human-AI co-creation of music (Ben-Tal *et al.*, 2020; Huang *et al.*, 2020).

In practice, we see different types of creativity given different types of notations and performers. (Cook, 2000) mentioned that "Eighteenth-century composers sometimes wrote down just the skeleton of what they intended, leaving the performer to flesh it out through figuration and ornamentation; twentieth-century composers, by contrast, generally try to specify what they want in far more detail." (Collins, 2011) described it well: "In a fraction of the time ... a competent composer could play/sing through several existing pieces of music, allow their musical brain to undertake the separating and reconnecting activities, and so devise a similarly successful new piece." In (Vieira & Schiavoni, 2020), it was concluded that pipe-and-filters, such as the ones in Pure Data and FAUST, can help more people to express themselves through music, in a simple, playful and graphical or textual way.

With the patterns we discussed, we can either retrieve these patterns for composers' inspirations or templates for new composition (Collins, 2011; Herremans & Chew, 2017). Performers can also learn these patterns to better remember or improvise on a piece of music, with transformations being the pipe-and-filters for compressing and manipulating the patterns. With patterns, it is one step closer for people or machines to obtain the same capacity as composers and performers with sophisticated musical patterning.

8.2.4 Music, health, and culture

Music therapy is part of regular healthcare and is the systematic use of specific musical interventions to improve healthcare outcome (Stegemann *et al.*, 2019). MIR is being considered for many applications in this domain.

The analysis of musical patterns has been used in analysing music improvised by patients with borderline personality disorders (Foubert *et al.*, 2017). (Baron-Cohen, 2020) describes the importance of pattern for people on the autism spectrum. As we listen to music of other cultures to understand those (sub)cultures, in (Cook, 2000), it was also noted that music therapy is where music communicates across the cultural barrier of mental disorders. We believe that more quantitative approaches to analysing musical patterns and transformations can bring additional insights for music therapy.

8.3 Looking Back

Looking back at our thesis statement:

Patterns, as a type of highly subjective and ubiquitous abstraction, have important connections to transformations, which can be explored compositionally using a functional programming language to reveal implications for designing musical pattern discovery algorithms.

In Figure 8.1, we show a summary of our transformation-based model for patterns and their repetitions and variations. Hopefully, the reader agrees that we have substantiated the statement with our chapters that review, implement, and analyse patterns and transformations. Although we do not have an ultimate solution to all the problems we have encountered, we have made progress in extending the available models and tools.

In the rest of the section, we will reflect on several significance and limitations of our work. In finishing, we will discuss work done in parallel to ours that employs similar thinking.

Significance

With each of our small significant results scattered over the chapters, we reiterate and integrate a few in this last chapter. Regarding patterns and transformations, for example, we hope to have captured and made use of these abstractions for understanding humans' and algorithms' approaches to musical pattern discovery. Regarding Haskell, we demonstrated that, with an appropriate language to express

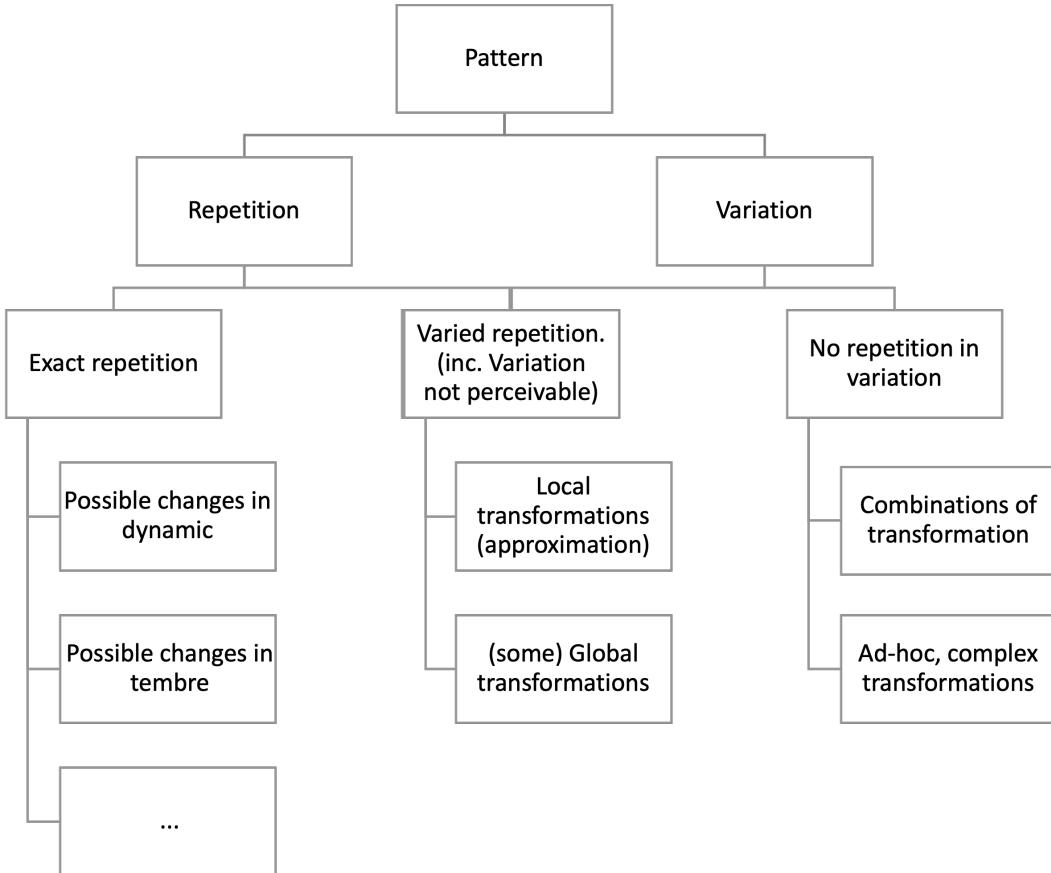


Figure 8.1: Combining pattern repetition or variation with transformations. On the left, we list the aspects we did not consider for exact repetition. In the middle, for varied repetition and non-perceivable variations, we devised local (approximation) and global transformations (e.g. transposition, rhythmic-only repetitions). The "some" in parenthesis points to a personalised model where some people might perceive transposition as variation or exact repetition, while others might regard them as completely different from the original material (people with perfectly-pitched or amusia). On the right, we list two circumstances where one might not be able to relate two pattern occurrences completely.

and communicate computational ideas, one could obtain a flexible environment to create and analyse music. As the perceived brittleness of algorithms often lies in how humans and algorithms succeed at a task differently, how to truly instill robustness into the models is closely related to how to bridge between humans and algorithms.

We also advocated for the realisation that benchmark evaluation does not equal impact. Faithful reproduction of the reference data is often not straightforward and not the goal. Instead, by trying to align algorithmic output and human reference, we wish to explore what kind of machines our minds are, and when is one more useful than the other. A related question is that of "what does it mean to be more human-

like when it comes to recognising pattern?" Our answer points to the organisation of exact repetition with a few transformations.

It is not our ambition to rewrite music analysis, pattern discovery, and functional programming—indeed, we only made small steps. Rather, we hope to have provided some new insights and perspectives on the pattern discovery task in music using a functional programming approach.

Limitations

We have discussed detailed limitations of our work in the individual chapters. We will not repeat them here but look at the limitations in four directions.

First, beyond the scope of this dissertation, there are many related methodologies. Although we cannot detail them all, we acknowledge that some of them raise important issues about the nature of patterns, transformations, and music. For instance, we have not considered any Bayesian aspects, which could be used in causal modelling and averaging models.

Second, in our efforts to connect two research areas, MIR and FP, we also only have limited success. We used knowledge from FP and PL for investigating musical patterns, but it is possibly more challenging to integrate the output of this research into providing new material for PL research. Having different research traditions and central research questions, it is hard for the connection between programming and art to be completely symmetrical: it is rarer for analysis in music to contribute to one of the central research topics in computational research than the other way around.

Third, the music data we used in this dissertation is limited, but the world teems with music of different types. Using monophonic MIDI datasets, we will never be able to analyse interesting music from many cultures. As described in (Cook, 2000), "in Indian and Chinese music it is often the notes between the notes, so to speak, that are responsible for the effect of the music." Another example is John Cage's 4'3", leaving aside the argument about whether it is music, our discussion does not extend to patterns from an empty space.

Last, we have taken a general view of patterns and more specific ones for transformations. When generality waxes, the discriminative power wanes, and vice versa. In general, it remains a challenge to both capture the fully open-ended nature of and the purposeful aspects of human pattern finding.

Convergence of other similar research

Independently of our work, we came across several recent papers using transformations and synthetic data for analysis and composition (Hunt *et al.*, 2019, 2020; Hunt, 2020; Silva *et al.*, 2020).

Using artificially repeated notes, (Silva *et al.*, 2020) used a range of methods, including **Convolutional Neural Network (CNN)** and **RNN**, to identify patterns, which is closely related to the synthetic data approach in Chapter 4.

(Hunt *et al.*, 2019) splits a piece of music into small segments based on bar lines and compare different transformations of the bars, including duplication, transposition, retrograde, inversion, retrograde-inversion, rotate left or rotate right, note-mapping arpeggiation. An application, IGME, was created for this. Using IGME, they found that classical and Bach contain more transposed and arpeggiated parts than any other datasets. Jazz has slightly more representability than classical but less than pop and rock. Video game music scores the highest overall: on average 62.79% of the music clips in the music are repeats of previous clips. Algorithmically generated songs also show different numbers to human-composed music. They were either too structureless or repetitive. Although using a different methodology, it is reassuring that our results do not contradict each other. They could not find retrograde, inversion, rotation with their approach and data, and we suspect that this is because they took a bar of music as the unit for comparison, which is not a common unit for inversion and retrograde.

Five types of musical repetition were proposed in (Collins, 2011): exact, with interpolation, transposed real, transposed tonal, and durational. We have considered them in this dissertation but implemented them differently. Furthermore, from (Collins *et al.*, n.d.), we see similar arguments to that we make in this dissertation:

The discovery and description of motives, themes, and repeated sections is complex, due to the myriad ways in which different occurrences may relate to one another, and the nature of the relationships between often-inexact occurrences. The tendency in music theory has been to address this complicated topic with broad brushstrokes, giving high-level and approximate summaries of form (e.g., ABA), and to use diverse definitions of motive and theme.

There are publications in other research areas that are also relevant to the discussions in previous chapters. For example, in the Abstraction and Reasoning Corpus (ARC) for measuring general intelligence (Chollet, 2019), detecting symmetry has been difficult for algorithms, with the best accuracy on Kaggle being at 20.6% (2021). In the research area of program synthesis, learning the transformations in

images has been investigated (Banburski *et al.*, 2020). Even Wittgenstein agrees with our approach to some extent: "understanding music consists in grasping the internal relationships between musical events" (Kaduri, 2006).

8.4 Looking Ahead

After acknowledging the limitations and biases in our work, we strive to continue being culturally and intellectually diverse and inclusive. While currently seen most in science fiction, the co-evolution between human thinking and machines is the trajectory we are currently on for the future. For us, to reiterate and make an addition to the future work we already proposed in previous chapters, we consider that this research could be extended in several directions.

Transformations

Transformations can be used in a range of other tasks in the future, such as classifying music from different areas and eras, creating a mapping between annotation reference dataset and concrete applications, and so forth. We can also use transformations to visualise and summarise the differences between every pair occurrences of a pattern, which informs about how these differences occur. Transformations can also be used to make intra-occurrences comparisons, as we discussed in Chapter 1.

Regarding transformations themselves, an extension would be to learn the transformations from data and combine this with more previously known transformations and structures in music. In fact, initial work on disentangling and discovering underlying transformations has emerged as an active research direction (Goyal *et al.*, 2021). The categorical modelling of transformations is another direction we have not explored. Checking for mathematical properties of the transformations can also potentially optimise and verify the results from our programs.

Music

The scope we set ourselves for this dissertation is monophonic symbolic music. We can certainly extend our datasets and consider polyphonic and audio data with multi-modal information, such as in combination with research from embodiment and dance: "Imagine, if you will, a new world order in which African approaches to reason pedagogy predominated in the American academy...No one would be

8 Conclusions and Future Work

granted a music degree who could not dance!" (Agawu, 2006). The important relations between music and our body that can be explored as mentioned in (Toiviainen *et al.*, 2010): "...we may use our bodily movements to help parse the metric structure of music".

Language

Because of the close relation between music and language, we can imagine some application in the language domain, too, especially the transformations in the context of NLP. We can look for relations such as the different degrees between sentences and words such as "good", "better", "best", which can be viewed as a type of transposition in music. In storytelling, there are methods such as playing with timeline, namely starting with the last events, which can be viewed as retrograde. For inversion, we can simply think of describing something from the opposite perspective. There is bound to be more transformation with diverse semantics as in natural language.

Some research in NLP also confirms this idea. (Jindal & Liu, 2006) identifies four types of comparatives: nonequal gradable, equative, superlative, and non-gradable. In (Mikolov *et al.*, 2013), a new dataset for measuring syntactic performance, including comparatives, was proposed, and the tested algorithms identified less than 40% correctly. To our knowledge, there has been little research on the ordering of large scale storytelling.

Algorithms

For pattern discovery algorithms, the coverage of the algorithms could be extended. For example, if we look for algorithms that can switch contexts, we may look into multi-task learning methods, where each task concerns a subset of factors, which require representation learning to disentangle the underlying factors of variation. In this way, we can find shared underlying explanatory factors between tasks, as we proposed for musical pattern discovery.

We can also try to use what we learned about patterns and transformations in designing the algorithms. In neural networks, for example, transformations may help with the task of data augmentation and the current challenges such as learning with fewer labelled samples, learning to reason, learning to plan complex action sequences, and so forth.

Synthetic data is another important direction going forward. There have been rapid developments and promising results in the **ML** field regarding using synthetic

data for both training and testing data for algorithms. Expanding on the example we have shown in Section 5.3.4, synthetic data can be created by more transformations and their combinations. Other simulation methods can also extend the possibilities to control the structures of patterns, such as data degradation processes with different degrees of severity (McLeod *et al.*, 2020). We could also combine other generative models to create synthetic data, and develop a feedback loop between the analytical and generative algorithms.

Annotations

For gathering annotations, the three experiments we reported can be refined and put into a more controlled environment. We can scale up the participants and also consider gamification for this purpose. A more diverse range of music can be annotated. The annotations can be collected with different concrete tasks in mind as well.

There is a problem, however, that although we know tasks such as compression and prediction, we do not know what "all the tasks" is. Some even argue that there will never be enough annotated data to train all the models for all the tasks we need to perform (Roth, 2017). If these views turn out to be true, we might need to switch away from using data to supervise the training of algorithms, and instead adopt ideas from weak- and self-supervision methods. Even so, we believe that annotation data would still be a valid starting point for developing algorithms.

Functional programming

For more work on the functional programming side, support and maintenance of the Pattrans package should be made. Type extensions, the order of the transformations, the selection of the prototype pattern, and analysis for other datasets can be further integrated. In combination with the synthetic data mentioned above, creating an expressive DSL for generating the desired data is a promising direction as well.

Appendices

Appendix A Datasets

In this appendix, we introduce the dataset we use in this dissertation. When we use the word "music", by default, we mean these dataset we investigated, unless explicitly disambiguated for a more general discussion. As put in (Cook, 2000), "the concept of 'music' was firmly rooted in a specific corpus of musical works, and through that in a specific time and place."

A.1 JKU-PDD

JKU-PDD, as the publicly available dataset from the MIREX task, has been used in evaluating musical pattern discovery algorithms. Compiled by MIR researchers, it contains one piece each by Bach, Mozart, Beethoven, Gibbons, Chopin, and 26 patterns and 105 occurrences annotated by experts. A list of the peace and the dates of the composers can be seen in Table A.1.

Composer	Dates	Piece
Orlando Gibbons	1583–1625	"The silver swan"
Johann Sebastian Bach	1685–1750	Fugue in A minor, BWV 889
Wolfgang Amadeus Mozart	1756–1791	Minuet from Piano Sonata in E major, K. 282
Ludwig van Beethoven	1770–1827	Scherzo from Piano Sonata in F minor, op. 2, no. 1
Frédéric Chopin	1810–1849	Mazurka in B minor, op. 24, no. 4

Table A.1: A summary of the pieces in the JKUPDD dataset.

Although the pattern occurrences are not annotated exhaustively (Meredith, 2015), it has been widely used for evaluating musical pattern discovery algorithms. According to (Collins *et al.*, n.d.), the annotations are constructed from three sources: Barlow and Morgenstern 1948, Schoenberg 1967, and Bruhn 1993. Some annotations were revised with added annotations for Gibbons' "The sliver swan". For example, because Barlow and Morgenstern 1948 is intended as a comprehensive yet concise companion for the classical music enthusiast, themes that are longer than the width of the page was curtailed for the sake of brevity. These are lengthened back to their musically appropriate length in the dataset.

When comparing annotations of the same piece across sources, it is quite common to find the sources in agreement. Although the sources have not gone through a inter-annotator agreement analysis, such a level of agreement is encouraging.

We use the monophonic version of this dataset, and to monophonise the music, the procedures of the clipped-skyline approach were followed as described below. From the polyphonic version, the slipped-skyline algorithm outputs the highest note at each unique onset with two scenarios. First, if the current highest note is still sounding when a new lower note begins, the new lower note is ignored. Second, if the current highest note is still sounding when a new higher note begins, the new higher note is included in the output, and the previous note's duration is clipped in time.

A.2 MTC-ANN

MTC-ANN is a Dutch folk song dataset (van Kranenburg *et al.*, 2016). The **MTC-ANN** dataset has been used for studying oral tradition in Dutch folk songs (van Kranenburg *et al.*, 2016), and the annotations consider inter-opus pattern occurrences. Inter-opus occurrences are not considered within one song, but between the songs within the same tune family.

Tune family is a concept in ethnomusicology that groups together tunes sharing the same ancestor in the process of oral transmission (Boot *et al.*, 2016). Oral transmission plays a significant role in folk music. Through this often imperfect communication process, certain parts of melodies remain stable, variations are created, repeated patterns emerge (Janssen, 2018). Formulated in ethnomusicological studies, the concept of tune family describes the structures in this stream of transformations: folk songs that are supposed to have a common ancestor in the process of oral transmission are grouped into a tune family (Cowdery, 1984). Local structures within the melodies, namely characteristic motifs, or prominent, nonliterally repeated patterns, are detected to be useful in determining music similarity and classifying tune families (Cowdery, 1984). Subsequently, in an annotation study on the influence of different musical dimensions on human similarity judgements of melodies belonging to the same tune family, repeated patterns between melodies turned out to play the most important role for similarity amongst all considered musical dimensions (Volk & Van Kranenburg, 2012). Therefore, algorithms which can extract these repeated patterns automatically that would be useful for tune family classification.

During the making of **MTC-ANN**, three experts were asked to annotate the prominent patterns in each song which best classify the song into one of 26 tune families.

The dataset consists of 360 Dutch folk songs with 1657 annotated pattern occurrences.

A.3 HEMAN and its different versions

As we introduced in Chapter 3, we use the same music material for three experiments, with the first one that initiated the others being called HEMAN. In the datasets, we observe that disagreement amongst annotators is common.

Only one of these experiments have the pattern-occurrence hierarchy, and that is dataset gathered by ANOMIC. We therefore call this dataset gathered by ANOMIC the HEMAN dataset in Chapter 7.

This HEMAN dataset contains 2763 annotations of pattern occurrences from 26 participants in at-home self-paced listening experiments. We separate the dataset into two subsets:

- HEMANLow, which is a sub-dataset of HEM, consists of the responses of participants with a self-rated musical background score < 5 on a scale of 1-10
- HEMANHigh is the complement of HEMANLow in HEMAN, and therefore consists of annotations of participants with a higher self-rated musical background.

A.4 Other datasets

We used two other music datasets in the MIDI format. The EuroVision dataset contains 200 transcribed songs from the EuroVision song contest over the course of 50 years (“The AI song contest”, 2020). The Jazz dataset contains jazz solos taken from the Omnibook compilation (Baker *et al.*, 2016). There are no pattern annotations available for the Eurovision and Jazz datasets. We will only use the corpora themselves to extract patterns using the algorithms.

Appendix B Introduction to Haskell

In this appendix, we provide a condensed introduction to Haskell for people who perhaps know Haskell from a long time ago or in an elementary capacity and could use some help to brush up on their memory. We will cover the following topics using examples from Haskell code:

- type
- type class
- pattern matching
- currying
- higher-order function
- lambda expression
- list comprehension
- recursion
- functor, applicative, monad operators

For more beginner friendly introductions, there are many great references such as (Hudak & Quick, 2018; Hutton, 2016; Lipovaca, 2011).

B.1 Defining type, type class

```
data NameOfCollection = Element1 | Element2 deriving  
    (Functionalities)
```

To unpack the components of the Haskell syntax, we try to use naming that is as close to the semantic meaning of those components as possible. Different functionality can be derived from type classes. As we do not aim to give an in-depth introduction to Haskell, we refer the readers to (Hutton, 2016) for the definition of type classes, and treat them as a way to define desired conditions and behaviours of the types. By explicitly writing the types in this way, one gets the benefit of compiler-level type checking.

B.2 Function, pattern matching, lambda expression, currying

In this section, we demonstrate how to define a function and how to use pattern matching. In Haskell, functions can be expressed compactly.

```
egfunctionName :: TypeClass => InputType1 -> InputTypeN ->
  ↵ ResultType
egfunctionName _ input = patternMatchResult1
egfunctionName input1 input2 = patternMatchResult2
```

Lambda expressions can also be used to define functions.

```
functionWithOneVariable = (\inputVariable -> manipulation
  ↵ inputVariable)
functionWithTwoVariables = (\inputVar1 -> (\inputVar2 ->
  ↵ manipulation inputVar1 inputVar2))
add = (\x -> (\y -> x + y))
add = (\x y -> x + y) -- syntax sugar of the above line
```

B.3 Currying and higher-order function

Currying, a process of making a function that takes multiple parameters into a sequence of functions each taking a single parameter, can also be used in defining functions. For example, in mathematics, we are used to thinking that add is a operator/function, but notice that (add 1) and (add 1 2) can also be functions, depending on the type and definition of add.

```
add3 x y z = x + y + z

add3 :: Num a => a -> a -> a -> a
add3 2 :: Num a => a -> a -> a
add3 2 1 :: Num a => a -> a
```

B.4 List comprehensions and laziness

List comprehension is one of the most powerful tools Haskell provides.

```
list :: [Int]
list = [1,2,3]
list = [1..3]
list = [x | x <- [1..3]]
-- three different ways of defining the same list

list = [f x | x <- [2,4..]]
-- laziness, another feature of Haskell, enable us to describe the
  ↵ structure but does not evaluate, therefore model the concept
  ↵ of infinity, e.g. all even numbers.

> [(+) x x | x <- [1..3]]
```

```

> [2,4,6]
> [(x,y) | x <- [1..3], y <- [1,1,1]]
> [(1,1),(1,1),(1,1),(2,1),(2,1),(2,1),(3,1),(3,1),(3,1)]
> [(+) x y | x <- [1..3], y <- [1,1,1]]
> [2,2,2,3,3,3,4,4,4]
-- manipulate lists with compact syntax

```

B.5 Recursion

In this section, we demonstrate recursion together with laziness using Fibonacci numbers.

```

fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
      -- a partial simulation of the recursion
> x = [1,1]
> zipWith (+) x (tail x)
[2]
> x = [1,1,2]
> zipWith (+) x (tail x)
[2,3]
> x = [1,1,2,3]
> zipWith (+) x (tail x)
[2,3,5]
> x = [1,1,2,3,5]
> zipWith (+) x (tail x)
[2,3,5,8]
-- another way to compute Fibonacci numbers using scanl -- an
-- accumulative function
> scanl (+) 1 [1,2,3,4,5]
[1,2,4,7,11,16]
fibs = 0 : scanl (+) 1 fibs
-- another way to compute Fibonacci numbers using fix -- three
-- different ways of encoding recursion
fix f = let {x = f x} in x
fix f = xs where xs = f xs
fix f = f (fix f)
fibs = fix ((0:) . scanl (+) 1)

```

B.6 Functor and more

Functors are everywhere in Haskell. Originally from category theory, we will see its categorical meaning in Appendix C.

Applicatives and monads are useful constructs, too, which are functors with additional constraints. We show the usage of functors, applicatives, and monads by giving examples of how the corresponding infix operators can be used with lists, which are themselves a type of functor. For a more detailed introduction, we refer the reader to (Hutton, 2016).

```

(<$>) :: Functor f => (a -> b) -> f a -> f b
(<$>) :: Functor f => (InputType -> OutputType) -> f InputType ->
    f OutputType
    -- Taking list as an example, this operator becomes
map :: (a -> b) -> [a] -> [b]

(<*>) :: Applicative f => f (a -> b) -> f a -> f b
(<*>) :: Applicative f => f (Input -> Output) -> f Input -> f
    Output
    -- examples when f is List, both expressions give:
    [3,4,5]

> (+2) <$> [1,2,3]
> map (+1) [1,2,3]
    -- examples when f is List, both expressions give:
    [3,4,5,5,6,7]

> [(+2),(+4)] <*> [1,2,3]
> [f b | f <- [(+2),(+4)], b <- [1,2,3]]
    -- both expressions give
    [3,4,5,5,6,7]

> (+) <$> [1,2,3] <*> [1,2,3]
> [(+)] <*> [1,2,3] <*> [1,2,3]
> liftA2 (+) [1,2,3] [1,2,3]
    -- the function liftA2 lift a binary function to actions: (a -> b
    -- -> c) -> f a -> f b -> f c
    -- all three expressions give
    [2,3,4,3,4,5,4,5,6]

(>>=) :: Monad m => m a -> (a -> m b) -> m b
(<=<) :: Monad m => (b -> m c) -> (a -> m b) -> a -> m c
    -- the bind and Kleisli operator of Monad

    -- creating a list of certain ranges using monad
> r x = [1..x]
> [3,5] >>= r
    [1,2,3,1,2,3,4,5]

    -- how this is different with the functor's map
> map r [3,5]
    [[1,2,3],[1,2,3,4,5]]

    -- streamlining computation using sequence :: Monad m => [m a] ->
    -- m [a]
> r = sequence [(+1), (+2)]
> [1,2] >>= r
    [2,3,3,4]

    -- another way to streamline computation of two functions f and g
> f x = [x, x + 1]
> g x = [x * x]

```

```
> t = g <=< f
> t 10
[100,121]
```


Appendix C Introduction to Category Theory

At the core of our implementation in Chapter 6 is the concept of contravariant bifunctor. In this appendix, we give the definitions of relevant concepts in category theory. Starting with the definition of a category, we proceed to giving the definition of functor, contravariant functor, and finally, the contravariant bifunctor. Finally, we give examples of constructs in category theory using Haskell code.

Definition C.0.1. A *category* \mathbf{C} consists of

- a collection of **objects**: A, B, C, \dots
- a collection of **arrows** or **morphism**: f, g, h, \dots
- each arrow f has a domain $\text{dom}(f)$ and a codomain $\text{cod}(f)$ consist of objects. If $\text{dom}(f) = A$ and $\text{cod}(f) = B$, we write $f : A \rightarrow B$ or f_{AB} , and call the collection of arrows a **hom-set** $\text{hom}(A, B)$,

we need

- $\forall f : A \rightarrow B$ and $g : B \rightarrow C$, there is an arrow $g \circ f : A \rightarrow C$,
- an arrow $1_A : A \rightarrow A$ for every object A of \mathbf{C} ,

such that

(**Associative law**) $\forall f : A \rightarrow B, g : B \rightarrow C, h : C \rightarrow D$, we have $h \circ (g \circ f) = (h \circ g) \circ f$,

(**Identity law**) for every $\forall f : A \rightarrow B$ we have $f \circ 1_A = f = 1_B \circ f$.

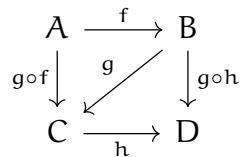


Figure C.1: Associative law.

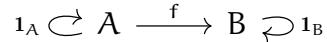


Figure C.2: Identity law..

Definition C.0.2. Given two categories \mathbf{C} and \mathbf{D} , a **functor** $F : \mathbf{C} \Rightarrow \mathbf{D}$ is a pair of functions assigning objects and arrows in \mathbf{C} to objects and arrows in \mathbf{D} correspondingly, such that,

- a covariant functor maps $f: A \rightarrow B$ in \mathbf{C} to $F(f): F(A) \rightarrow F(B)$ in \mathbf{D} .
- a contravariant functor maps $f: A \rightarrow B$ in \mathbf{C} to $F(f): F(B) \rightarrow F(A)$ in \mathbf{D} .
- preserve identities and composition, $F(1_A) = 1_{F(A)}$ and $F(f \circ g) = F(f) \circ F(g)$ for a covariant functor, and $F(f \circ g) = F(g) \circ F(f)$ for a contravariant functor.

Definition C.0.3. Given two categories \mathbf{C} and \mathbf{D} , the **product** of \mathbf{C} and \mathbf{D} is the category $\mathbf{C} \times \mathbf{D}$, with

- the objects C and D from the respective categories forming pairs (C, D) ,
- morphisms $f: C \rightarrow C'$ and $g: D \rightarrow D'$ forming $(f, g): (C, D) \rightarrow (C', D')$

such that the composition is componentwise: $(f, g) \circ (h, k) = (f \circ h, g \circ k)$

Definition C.0.4. Given three categories $\mathbf{C}, \mathbf{D}, \mathbf{A}$, and the product $\mathbf{C} \times \mathbf{D}$, a **bifunctor** is $F: \mathbf{C} \times \mathbf{D} \rightarrow \mathbf{A}$.

In Haskell, one simple functor is `Maybe`:

```
data Maybe a = Nothing | Just a

instance Functor Maybe where
  fmap :: (a -> b) -> (Maybe a -> Maybe b)
  fmap _ Nothing = Nothing
  fmap f (Just x) = Just (f x)
```

Code Snippet C.1: An example of Functor `Maybe`

In Chapter 6, the generalised underlying structure of our checker is defined as below:

```
class ContravariantBifunctor p where
  contraBimap :: (c -> a) -> (d -> b) -> p a b -> p c d
  contraBimap f g = contra1 f . contra2 g

  contra1 :: (c -> a) -> p a b -> p c b
  contra1 f = contraBimap f id

  contra2 :: (d -> b) -> p a b -> p a d
  contra2 g = contraBimap id g

instance ContravariantBifunctor Check where
  contraBimap f g p = MkCheck (\ x y -> (f x <=> g y) p)
```

Code Snippet C.2: Defining contravariant bifunctor in Haskell

Figure C.3 and Figure C.4 diagrammatically illustrate checkers of different types that can be related using contravariant bifunctor.

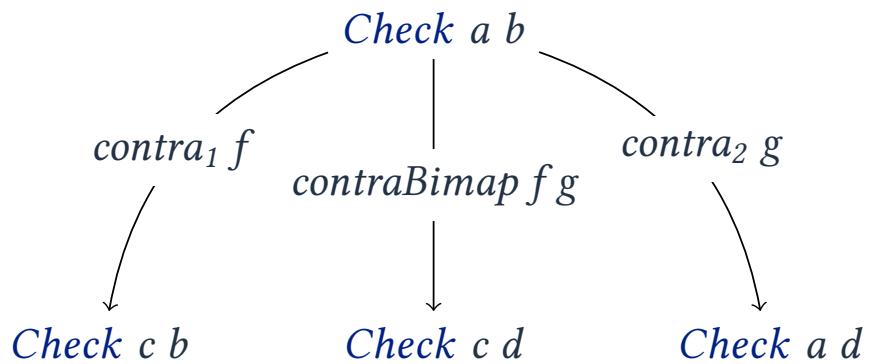


Figure C.3: Checkers forming the contravariant bifunctor.

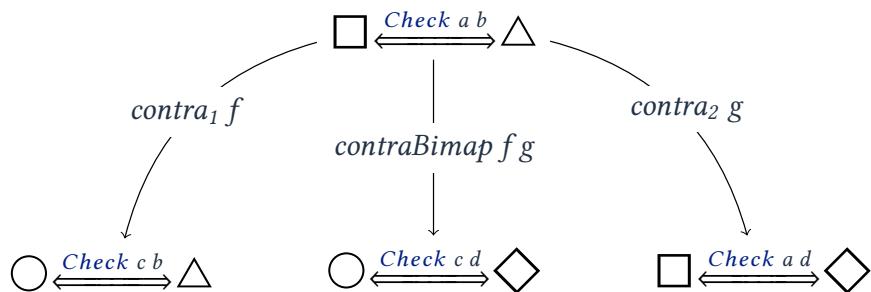


Figure C.4: Using shapes to illustrate types in Figure C.3.

Appendix D Pattrans Package

In this appendix, we provide the detail of the Pattrans package, which we use as a main tool for this thesis.

D.1 Basic types and functions

In addition to the types we introduced in Section 6.3.1, we have the more types regarding music primitives. These types and more accompanying utility functions are also constructed as follows.

```
type ScaleDegree = Int
type Octave     = Int
type ScaleType  = [Interval]

major, harmonicMinor, melodicMinor :: ScaleType
major      = [0,2,4,5,7,9,11]
melodicMinor = [0,2,3,5,7,9,11]
harmonicMinor = [0,2,3,5,7,8,11]

-- | A piece of music is a huge pattern.
type MusicPiece = Pattern

-- | Negate the values of a numeric list.
inverse :: Num a => [a] -> [a]
inverse = fmap negate

-- | The base pitch of a pattern (the pitch of its first note).
-- e.g. basePitch [(25,1), (27,2), (25,2.5)] = Just 25
basePitch :: Pattern -> Maybe MIDI
basePitch (Note _ m:_ ) = Just m
basePitch []            = Nothing

-- | The (real) rhythmic structure of a pattern.
-- e.g. onset [(25,1), (27,2), (25,2.5)] = [1, 2, 2.5]
onsets :: Pattern -> [Time]
onsets = sort . fmap ontime

contour :: Pattern -> [Contour]
contour p = map signum $ intervals p

-- | Normalized (relative) rhythmic structure of a pattern.
-- e.g. normalRhythm [(A,2), (C#,6), (Eb,8), (B,1), (A,2)] = [1,
--                   3, 4, 1/2, 1]
normalRhythm :: Pattern -> [Time]
normalRhythm = normalizeTime . rhythm
```

```

where
-- | Convert times to ratios wrt the first time unit used.
-- e.g. normalizeTime [2, 6, 8, 6, 1, 2] = [1, 3, 4, 1/2, 1]
normalizeTime :: [Time] -> [Time]
normalizeTime (tt : ts) = 1 : ((/ tt) <$> ts)
normalizeTime [] = []

-- | Translate a note horizontally (in time).
-- e.g. translateH (-0.5) [(25,1), (27,2), (25,2.5)] = [(25,0.5),
--   (27,1.5), (25,2)]
translateH :: Time -> Note -> Note
translateH dt (Note tInit m) = Note (tInit + dt) m

-- | Translate a note vertically (in pitch).
-- e.g. translateV (-20) [(25,1), (27,2), (25,2.5)] = [(5,1),
--   (7,2), (5,2.5)]
translateV :: Interval -> Note -> Note
translateV dm (Note tt mInit) = Note tt (mInit + dm)

```

D.2 Approximation

As mentioned in Chapter 6, we give a complete version of the approximation function in Pattrans. Provide detailed annotation for navigating through the definition. Noticed that the function `approxEq` is refactored to a different version to be in parallel with `approxEq2`.

```

approxEqWith :: forall b. (Show b, Num b, Eq b)
=> (b -- the element to delete
     -> [b] -- the initial list
     -> Int -- maximum elements to ignore
     -> Maybe (Int, [b]) -- * Nothing, if there was no
                           -- deletion
                           -- * Just(# of ignored, tail),
                           --   -- otherwise
     )
     -- ^ function that deletes an element from a list,
     -- possibly
     -- reducing (summing) consecutive elements to be
     -- equal to the
     -- element being deleted
     -> ApproxCheck [b]
approxEqWith del1
| ?p == 1.0 = equal -- short-circuit for faster results
| otherwise = Check go
where
  go :: (?p :: Float) => [b] -> [b] -> Bool
  go xs' ys' =
    let [xs, ys] = sortOn length [xs', ys']
    [n, m] = length <$> [xs, ys]
    maxIgnored = floor $(1 - ?p) * fromIntegral n
    maxAdded = floor $(1 - ?p) * fromIntegral m
    in del ys xs (maxIgnored, maxAdded)
  del :: [b] -> [b] -> (Int {-ignored-}, Int {-added-}) -> Bool

```

```

del ys []      (maxI, maxA) = 0           <= maxI && length ys <=
  ↵ maxA
del [] xs      (maxI, maxA) = length xs <= maxI && 0           <=
  ↵ maxA
del ys (x:xs) (maxI, maxA)
  -- surpassed the limits, abort
  | maxI < 0 || maxA < 0
  = False

  -- no more additions/ignores allowed, resort to simple
  -- equality
  | maxI + maxA == 0
  = ys == x:xs

  -- found the prototype element in the occurrence (possibly
  -- adding elements)
  -- It could be the case that it's better to ignore it though,
  -- thus the second case
  | Just (maxA', ys') <- del1 x ys maxA
  , maxA' >= 0
  , maxA - maxA' <= maxLookahead
  = del ys' xs (maxI, maxA')
  -- `|| (maxI > 0 && del ys xs (maxI - 1, maxA))` -- too
  -- slow...

  -- did not find element, ignore if possible
  | maxI > 0
  = del ys xs (maxI - 1, maxA)

  -- did not find element and cannot ignore it, abort
  | otherwise
  = False

-- | First-order approximate equality of lists.
-- 
-- Check that two lists are approximately equal, wrt a certain
-- percentage.
-- A base pattern and an occurrence are approximately equal with
-- percentage `p` when:
--   1. The occurrence ignores  $(1-p)\%$  notes of the base pattern
--   2.  $(1-p)\%$  notes of the occurrence are additional notes (not
--      in the base pattern)
-- e.g. [A,C,F,A,B] (approxEq 80%) [A,C,G,A,B]
approxEq :: (Show a, Num a, Eq a) => ApproxCheck [a]
approxEq = approxEqWith del1
where
  -- does not reduce consecutive elements (first-order)
  del1 []      = Nothing
  del1 x (y:ys) maxA
    | maxA < 0 = Nothing
    | x == y   = Just (maxA, ys)
    | otherwise = del1 x ys $! (maxA - 1)

-- | The essential difference with first-order approximate
-- equality is the ability
-- to equate consecutive elements with their sum, hence the
-- Ord/Num constraint.

```

```

approxEq2 :: (Show a, Ord a, Num a, Eq a) => ApproxCheck [a]
approxEq2 = approxEqWith del1
where
  -- reduces consecutive elements (second-order)
  del1 _ [] _ = Nothing
  del1 x (y:ys) maxA
    | maxA < 0 = Nothing
    | x == y = Just (maxA, ys)
    | Just i <- findIndex 0 x (y:ys) maxLookahead -- limitation:
      ↳ fixed look-ahead
    , maxA >= i
    = Just (maxA - i, snd $ splitAt i ys)
    | otherwise
    = del1 x ys $! (maxA - 1)

  findIndex i 0 _ _ = Just i
  findIndex _ _ _ 0 = Nothing
  findIndex _ _ [] _ = Nothing
  findIndex i acc (y:ys) maxAc
    | acc >= y = findIndex (i + 1) (acc - y) ys (maxAc - 1)
    | otherwise = Nothing

```

D.3 Query

As introduced in Chapter 7, we use Euterpea for defining prototype patterns for query. We make this possible with the functions below.

```

import Euterpea.Music as Export hiding (Rest, Note,
  ↳ pitch)
import qualified Euterpea.Music as M

-- | Convert our Pattern datatype to Euterpea's music datatype.
patternToMusic :: Pattern -> Music AbsPitch
patternToMusic = line . fmap convert . withDurations
where
  withDurations :: Pattern -> [(MIDI, Time)]
  withDurations ps = zip (pitch ps) (durations ps)

  convert :: (MIDI, Time) -> Music AbsPitch
  convert (m, tt) = Prim $ M.Note ((toRational tt / 4) )
    ↳ (fromInteger m)

-- | Convert from Euterpea's music datatype to our pattern
-- datatype.
musicToPattern :: ToMusic1 a => Music a -> Pattern
musicToPattern = withDurations . convert . fmap (absPitch . fst) .
  ↳ toMusic1
where
  convert :: Music AbsPitch -> [Either Time Note]
  convert (n :+: ns) = convert n ++ convert ns
  convert (n ::= _) = convert n
  convert (Modify _ n) = convert n
  convert (Prim prim) =
    case prim of
      M.Rest r -> [Left $ fromRational r]

```

```

M.Note dr p -> [Right $ Note (fromRational dr) (toInteger
  ↵  p)]

withDurations :: [Either Time Note] -> [Note]
withDurations = snd . foldl go (0.0, [])
  where go :: (Double, [Note]) -> Either Time Note -> (Double,
    ↵  [Note])
    go (acc, ns) (Left tt)           = (acc + tt, ns)
    go (acc, ns) (Right (Note tt m)) = (acc', ns ++ [Note
      ↵  acc m])
    where acc' = acc + tt

-- | Given a datatype that can be converted to Euterpea's core
-- Music datatype,
-- one can subsequently convert that to get a musical pattern.
instance ToMusic1 a => ToPattern (Music a) where
  toPattern _ = musicToPattern

```


Appendix E List of Publications

Some of the work contained in this dissertation has appeared in the following publications:

- An earlier version of the research reported in chapter 3 was also reported in:

Investigating Musical Pattern Ambiguity in a Human Annotated Dataset, IY Ren, O Nieto, HV Koops, A Volk, W Swierstra. International Conference on Music Perception and Cognition/European Society for the Cognitive Sciences of Music. Graz, Germany, 2018.

- Early work into feature and PCA analysis of musical patterns was published in:

Feature Analysis of Repeated Patterns in Dutch Folk Songs using Principal Component Analysis. IY Ren, HV Koops, D Bountouridis, A Volk, W Swierstra, R Veltkamp. Folk Music Analysis, Thessaloniki, Greece, 2018.

- Part of chapter 5 is based on:

Analysis by Classification: A Comparative Study of Annotated and Algorithmically Extracted Patterns in Symbolic Music Data. IY Ren, A Volk, W Swierstra, R Veltkamp. International Society of Music Information Retrieval, Paris, France, 2018.

In Search of the Consensus among Musical Pattern Discovery Algorithms. IY Ren, HV Koops, A Volk, W Swierstra. International Society of Music Information Retrieval, Suzhou, China, 2017.

A Computational Evaluation of Musical Pattern Discovery Algorithms

- An earlier version of Pattrans and analysis presented in chapter 6 and 7 was published in:

Orestis Melkonian, Iris Yuping Ren, Wouter Swierstra, and Anja Volk. "What constitutes a musical pattern?". In Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM 2019). ACM, New York, NY, USA, p. 95-105, 2019.

Glossary

accuracy	the proportion of true positive and true negative predictions in the context of all predictions. 219
ambiguity	the same data that is open to multiple interpretations. An ambiguous sentence or phrase is also known as a "crash blossom"..... 5, 6
augmentation	lengthening of note durations—time scaling, slowing down. Alternatively, widening of an interval by a chromatic semitone. Opposite of diminuation ... 180, 183, 257
chromatic transposition	movement of note pitches by a fixed interval—vertical translation..... 180–183
closed pattern	a pattern that has no superset (longer from left or right) that has the same support (number of repetitions). 61, 63
cross-entropy	a way to quantify the difference between two probability distributions..... 219
cross-validation	partitioning the training dataset and withholding partition(s) during multiple instances of training. Often used to estimate how well a model will generalise to new data. 131
diminuation	shorting of note durations—time scaling, speeding up. Alternatively, narrowing of an interval by a chromatic semitone. Opposite of augmentation 180, 183, 257

Glossary

equivariance	a transformation is equivariant when the output changes with respect to the input, e.g. equivariant to translation means that a translation of input results in an translation of output. See also invariance 169 , 258
exact repetition	repetition of a pattern occurrence with exactly the same musical events—horizontal translation, transposition in time. ... 180–183
F_1 score	the harmonic mean of precision and recall. The harmonic mean can be expressed as the reciprocal of the arithmetic mean of the reciprocals. ... 60 , 90 , 128 , 146
ground truth	often used in the evaluation of algorithms in ML ; consists of a list of information to be known for being "true", and hence expected from the output of the algorithms. ... vii , 74 , 75 , 145
heuristic	a directional shortcut to solve problems or make decisions quickly and efficiently that can come from a wide variety of sources: from ad-hoc tricks to general domain knowledge , i.e. rule-of-thumb strategies and methods. Can be viewed as a type of bias. See (Romanycia & Pelletier, 1985)... 57 , 60
invariance	given a transformation, the output does not change with different input, e.g. invariant to translation means that a translation of input results in the same output. See also equivariance 170 , 258
inversion	a change in the direction of pitch intervals—horizontal reflection, e.g. an ascending scale becomes a descending scale. ... 180 , 183

maximal pattern	is a pattern that has no superset (longer from left or right) that is frequent (a pre-set number of repetitions).	62, 63
monad	inspired by category theory, in Haskell, a monad is used to incorporate various side-effects. Disambiguation.: this word can also refer to a musical construct.	10
precision	the proportion of true positive predictions in the context of all predictions.	90, 128
recall	the proportion of true positive predictions in the context of all reference. Also known as sensitivity.	90, 128
retrograde	reversal of a musical passage—vertical reflection.	180, 183
retrograde inversion	mirror note durations backwards and change the direction of pitch intervals —an example composition of transformations.	183
rhythmic-only repetition	a repetition of the rhythm of a pattern occurrence while permitting any pitch transformations other than exact repetition and chromatic transposition.	181, 182
rotation	Change the direction of pitch intervals and mirror note durations backwards—an example composition of transformations, different from a retrograde-inversion in its different first note.	183
self-similarity	a property that the whole is similar to a part in some capacity, e.g. fractals such as Mandelbrot set, coastline, and broccoli. Related to self-similarity matrix	260

Glossary

self-similarity matrix	with a metric of distance, one can measure the similarity of one element in a sequence with respect to all other elements and obtain a vector of distances. One can then obtain such vectors for every element in the sequence and arrange them into a matrix. Related to <i>self-similarity</i> in that one part is similar to another part of itself.	62, 259
subjectivity	interpreting the same data in multiple ways based on some intrinsic aspects of the observer. For example, the same musical event can trigger different conceptualisations of the music.	6
tonal transposition	movement of note pitches in scale degree—vertical translation.	180
transformation profile	consists of the proportions of transformations in a pattern group. Can be viewed as a vector with its dimensions corresponding to the percentages of each transformation type.	211, 212
viewpoint	depending on context, this word could mean a "musical feature" as described in Chapter 4, or simply the dictionary meaning of "an opinion or point of view".	40

Samenvatting

Het vinden van "patronen" is een van de meest voorkomende activiteit bij menselijke intellectuele inspanningen. Wat betekent het als mensen zeggen "Ik zie hier een patroon"? En wat betekent het als mensen in muziek hierop in muziek wijzen? Er zijn een paar mogelijkheden, afhankelijk van wie het zegt en met welk doel. We onderzoeken de meervoudige betekenissen van "patroon" en de methoden om ze te vinden. Met muziek als onze focus, verzamelen, visualiseren, vergelijken en gebruiken (exploiteren) we menselijke geannoteerde patronen en algoritmisch geëxtraherde patronen. We identificeren het belang van transformaties achter de patronen. Geïnspireerd door functioneel programmeren, gebruiken we de functionele taal Haskell om deze verbinding tussen patronen en transformatie te modelleren. Op het hoogtepunt van alle ideeën en onderzoek, verkennen we hoe we muzikale transformaties kunnen gebruiken om muzikale patroonvoorvalen te verbinden en te classificeren, waarbij we inzichten onthullen over een breed scala aan patronen van zowel algoritmische output als menselijke annotaties.

Curriculum Vitae

Iris Yuping Ren was born and grew up in Beijing, China, where she studied violin, computer science, mathematics, and astronomy through high school. Between 2009 and 2013, she completed a Bachelor of Science in the School of Mathematics and a Bachelor of Management Studies in the school of History and Culture at Shandong University, China. After taking additional modules in physics, she successfully applied to PhD programmes in mathematics and physics, but decided to start her academic career with the Erasmus Mundus programme in complex systems science. Under this programme, she then completed two Masters of Science at the University of Warwick, U.K., and École Polytechnique, France, with a thesis project on modelling symbolic music using complex network theory and analysing these networks using topological data analysis. In pursuit of her interests in computational methods for music, she entered the Electrical and Computer Engineering department at the University of Rochester. During this time in the U.S., she completed a Master's degree in Electrical Engineering, earned an Advanced Diploma in Violin at Eastman Community Music School, and participated in courses and research offered by the Department of Brain and Cognitive Sciences. Following this trajectory, in 2017, she started her PhD in computational music structure analysis using functional programming under the supervision of dr. Anja Volk, dr. Wouter Swierstra, prof. dr. Johan Jeuring, and prof. dr. Remco C. Veltkamp at the Department of Information and Computing Sciences at Utrecht University. Due to departmental changes, prof. dr. Gabriele Keller joined the supervisory team in 2019 in place of Johan. During her PhD, Iris served on the Faculty Council, Departmental Advisory Committee, and as a board member of the International Society of Music Information Retrieval. Other interests she explored during her time as a PhD candidate include the history and philosophy of science, repetitive strain injury, and diversity and inclusion.

References

- 2005:Symbolic Key Finding Results - MIREX Wiki. (2005).
- 2014:Discovery of Repeated Themes & Sections - MIREX Wiki. (2014).
- 2017:Discovery of Repeated Themes & Sections - MIREX Wiki. (2017).
- Aaron, S. (2016). Sonic Pi-performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, 12(2), 171–178.
- Acotto, E., & Andreatta, M. (2012). Between mind and mathematics: Different kinds of computational representations of music. *Mathématiques et Sciences Humaines. Mathematics and Social Sciences*, (199), 7–25.
- Agawu, K. (2006). Structural analysis or cultural analysis? Competing perspectives on the “standard pattern” of West African rhythm. *Journal of the American Musicological Society*, 59(1), 1–46.
- Agawu, K. (2014). *Music as discourse: Semiotic adventures in romantic music*. Oxford University Press.
- Alaa, A. M., van Breugel, B., Saveliev, E., & van der Schaar, M. (2021). How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. *arXiv preprint arXiv:2102.08921*.
- Allegraud, P., Bigo, L., Feisthauer, L., Giraud, M., Groult, R., Leguy, E., & Levé, F. (2019). Learning sonata form structure on mozart’s string quartets. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 2(1), 82–96.
- Amabile, T. M. (1988). A model of creativity and innovation in organizations. *Research in Organizational Behavior*, 10(1), 123–167.
- Andreatta, M. (2018). From music to mathematics and backwards: Introducing algebra, topology and category theory into computational musicology. *Imagine math 6* (pp. 77–88). Springer.
- Andrienko, N., Andrienko, G., Miksch, S., Schumann, H., & Wrobel, S. (2021). A theoretical model for pattern discovery in visual analytics. *Visual Informatics*, 5(1), 23–42.
- Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4), 319–342.
- Argentino, J., & Mackenzie, S. (2019). Transformations in serial music and language games. *Journal of New Music Research*, 48(2), 159–179.
- Arjonilla, F. J., & Ogata, T. (2017). General problem solving with category theory. *arXiv preprint arXiv:1709.04825*.
- Baez, J., & Stay, M. (2010). Physics, topology, logic and computation: A rosetta stone. *New structures for physics* (pp. 95–172). Springer.

References

- Baker, D. J., Rosado, A., Shanahan, E., & Shanahan, D. (2016). The role of idiomatism and affordances in bebop improvisation. *Proceedings of the 14th International Conference on Music Perception and Cognition (ICMPC)*, 127–130.
- Baldazzi, D., & Ghifary, M. (2016). Strongly-typed recurrent neural networks. *arXiv preprint arXiv:1602.02218*.
- Balke, S., Driedger, J., Abeßer, J., Dittmar, C., & Müller, M. (2016). Towards evaluating multiple predominant melody annotations in jazz recordings. *Proceedings of the 17th International Society for Music Information Retrieval (ISMIR)*, 246–252.
- Bamberger, J. S. (2000). *Developing musical intuitions: A project-based introduction to making and understanding music*. Oxford University Press.
- Banburski, A., Ghandi, A., Alford, S., Dandekar, S., Chin, P., & Poggio, T. (2020). *Dreaming with ARC* (tech. rep.). Center for Brains, Minds and Machines (CBMM).
- Baron-Cohen, S. (2020). *The pattern seekers: How autism drives human invention*. Basic Books.
- Beghetto, R. A., & Kaufman, J. C. (2007). Toward a broader conception of creativity: A case for "mini-c" creativity. *Psychology of Aesthetics, Creativity, and the Arts*, 1(2), 73.
- Ben-Tal, O., Tobias Harris, M., & Sturm, B. L. (2020). How music AI is useful: Engagements with composers, performers, and audiences. *Leonardo*, 1–13.
- Bergin, J., Eckstein, J., Manns, M., Sharp, H., Maraquardt, K., Chandler, J., Sipos, M., Völter, M., & Willingford, E. (2012). Pedagogical patterns: The pedagogical patterns project. *Pedagogical Patterns: Advice for Educators*, Joseph Bergin Software Tools.
- Bernstein, L. (1976). *The unanswered question: Six talks at harvard* (Vol. 33). Harvard University Press.
- Bertens, R., Vreeken, J., & Siebes, A. (2016). Keeping it short and simple: Summarising complex event sequences with multivariate patterns. *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, ACM SIGKDD, 735–744.
- Besheli, P. R. (2018). The pattern of patterns: What is a pattern in conceptual modeling? *Proceedings of the 12th International Workshop on Value Modeling and Business Ontologies (VMBO)*, 99–106.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Blacking, J. (1984). *What languages do musical grammars describe?* Leo S. Olschki Editore.
- Boden, M. A. (1996). Creativity. *Artificial intelligence* (pp. 267–291). Elsevier.
- Bomberger, W. A. (1996). Disagreement as a measure of uncertainty. *Journal of Money, Credit and Banking*, 28(3), 381–392.
- Boot, P., Volk, A., & de Haas, W. B. (2016). Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3), 223–238.
- Bountouridis, D. (2018). *Music information retrieval using biologically inspired techniques* (Doctoral dissertation). Utrecht University.

- Brady, E. (2017). *Type-driven development with Idris*. Manning Publications Company.
- Brand, J., Hailey, C. et al. (1997). *Constructive dissonance: Arnold schoenberg and the transformations of twentieth-century culture*. University of California Press.
- Brand, M. (1999). Pattern discovery via entropy minimization. *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics (AISTAT)*.
- Bregman, M. R., Patel, A. D., & Gentner, T. Q. (2016). Songbirds use spectral shape, not pitch, for sound pattern recognition. *Proceedings of the National Academy of Sciences*, 113(6), 1666–1671.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.
- Britannica, E. et al. (2013). *Britannica book of the year 2013*. Encyclopaedia Britannica, Inc.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Buteau, C., & Mazzola, G. (2000). From contour similarity to motivic topologies. *Musicae Scientiae*, 4(2), 125–149.
- Buteau, C., & Mazzola, G. (2008). Motivic analysis according to rudolph réti: Formalization by a topological model. *Journal of Mathematics and Music*, 2(3), 117–134.
- Calvo-Zaragoza, J., Castellanos, F. J., Vigliensoni, G., & Fujinaga, I. (2018). Deep neural networks for document processing of music score images. *Applied Sciences*, 8(5), 654.
- Cambell, E. (2010). *Boulez, music and philosophy*. Cambridge University Press.
- Cambouropoulos, E. (2001). Melodic cue abstraction, similarity, and category formation: A formal model. *Music Perception*, 18(3), 347–370.
- Cambouropoulos, E. (2006). Musical parallelism and melodic segmentation. *Music Perception: An Interdisciplinary Journal*, 23(3), 249–268.
- Campbell, D. T. (1960). Blind variation and selective retentions in creative thought as in other knowledge processes. *Psychological Review*, 67(6), 380.
- Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., & Duchi, J. C. (2019). Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*.
- Chiu, B., Keogh, E., & Lonardi, S. (2003). Probabilistic discovery of time series motifs. *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining ACM SIGKDD*, 493–498.
- Chollet, F. (2019). On the measure of intelligence.
- Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.
- Chuan, C.-H., Agres, K., & Herremans, D. (2020). From context to concept: Exploring semantic relationships in music with word2vec. *Neural Computing and Applications*, 32(4), 1023–1036.
- Clarke, R. T. (1934). The drum language of the tumba people. *American Journal of Sociology*, 40(1), 34–48.

References

- Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. *Proceedings of the 33th International Conference on Machine Learning (ICML)*, 2990–2999.
- Collins, T. (2011). *Improved methods for pattern discovery in music, with applications in automated stylistic composition* (Doctoral dissertation). The Open University.
- Collins, T., Arzt, A., Flossmann, S., & Widmer, G. (2013). SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 549–554.
- Collins, T., Flossmann, S., Arzt, A., & Widmer, G. (n.d.). A tutorial on pattern discovery: Auditioning, formalizing, and evaluating the discovery of motives, themes, and repeated sections in symbolic and audio representations of music.
- Colton, S. (2012). The painting fool: Stories from building an automated painter. *Computers and creativity* (pp. 3–38). Springer.
- Colton, S., Wiggins, G. A. et al. (2012). Computational creativity: The final frontier? *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, 12, 21–26.
- Conklin, D. (2002). Representation and discovery of vertical patterns in music. *International Conference on Music and Artificial Intelligence*, 32–42.
- Conklin, D. (2010). Discovery of distinctive patterns in music. *Intelligent Data Analysis*, 14(5), 547–554.
- Conklin, D. (2013). Antipattern discovery in folk tunes. *Journal of New Music Research*, 42(2), 161–169.
- Conklin, D., & Anagnostopoulou, C. (2001). Representation and discovery of multiple viewpoint patterns. *Proceedings of the 27th International Computer Music Conference (ICMC)*.
- Conklin, D., & Anagnostopoulou, C. (2006). Segmental pattern discovery in music. *INFORMS Journal on computing*, 18(3), 285–293.
- Conklin, D., & Anagnostopoulou, C. (2011). Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40(2), 119–125.
- Conklin, D., & Bergeron, M. (2008). Feature set patterns in music. *Computer Music Journal*, 32(1), 60–70.
- Conklin, D., & Maessen, G. (2019). Generation of melodies for the lost chant of the mozarabic rite. *Applied Sciences*, 9(20), 4285.
- Cook, N. (2000). *Music: A very short introduction*. Oxford University Press.
- Cooley, R., Mobasher, B., & Srivastava, J. (1997). Web mining: Information and pattern discovery on the world wide web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, 558–567.
- Costa, Y. M., Oliveira, L. S., & Silla Jr, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, 52, 28–38.
- Cowdery, J. R. (1984). A fresh look at the concept of tune family. *Ethnomusicology*, 28(3), 495–504.

- Crumley, C. L. (1995). Heterarchy and the analysis of complex societies. *Archeological Papers of the American Anthropological Association*, 6(1), 1–5.
- da Fontoura Costa, L., & Cesar Jr, R. M. (2009). *Shape classification and analysis: Theory and practice*. CRC Press, Inc.
- Dannenberg, F. K., Dannenberg, R. B., & Miller, P. L. (1984). Teaching programming to musicians.
- Davies, M. (2021). Constraining the AI solution space with biological principles.
- De Haas, W. B. (2012). *Music information retrieval based on tonal harmony* (Doctoral dissertation). Utrecht University.
- Deliège, I., & Wiggins, G. A. (2006). *Musical creativity: Multidisciplinary research in theory and practice*. Psychology Press.
- de Reuse, T., & Fujinaga, I. (2019). Pattern clustering in monophonic music by learning a non-linear embedding from human annotations. *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 761–768.
- Deutsch, D. (2019). *Psychology and music*. Psychology Press.
- Dibbets, J. (2002). *Interactions between science and art*. Elsevier Science.
- Dixon, S., Gouyon, F., & Widmer, G. (2004). Towards characterisation of music via rhythmic patterns. *Proceedings of the 5th International Society for Music Information Retrieval (ISMIR)*.
- Dolan, E. I. (2013). *The orchestral revolution: Haydn and the technologies of timbre*. Cambridge University Press.
- Dowling, W. J. (1972). Recognition of melodic transformations: Inversion, retrograde, and retrograde inversion. *Perception & Psychophysics*, 12(5), 417–421.
- Drabkin, W. (2001a). Motif.
- Drabkin, W. (2001b). Theme.
- Dunning, D. (2011). The dunning–kruger effect: On being ignorant of one's own ignorance. *Advances in experimental social psychology* (pp. 247–296). Elsevier.
- Ewell, P. (2020). Music theory's white racial frame. *Music Theory Online*, 26(2).
- Fallows, D. (2001). Head-motif.
- Ferguson, D. N. (1941). What is a musical idea? *Papers of the American Musicological Society*, 43–49.
- Finkensiep, C., Déguernel, K., Neuwirth, M., & Rohrmeier, M. (2020). Voice-leading schema recognition using rhythm and pitch features. *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*.
- Fiske, H. E. (1997). Categorical perception of musical patterns: How different is "different". *Bulletin of the Council for Research in Music Education*, (133), 20–24.
- Fitch, W. T. (2006). The biology and evolution of music: A comparative perspective. *Cognition*, 100(1), 173–215.
- Flexer, A., & Grill, T. (2016). The problem of limited inter-rater agreement in modelling music similarity. *Journal of New Music Research*, 45(3), 239–251.
- Floyd, R. W. (1967). Assigning meanings to programs. *Proceedings of the American Mathematical Society Symposia on Applied Mathematics*, 19, 19–31.

References

- Fong, B., Spivak, D., & Tuyéras, R. (2019). Backprop as functor: A compositional perspective on supervised learning. *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–13.
- Forte, A. (1959). Schenker's conception of musical structure. *Journal of Music Theory*, 3(1), 1–30.
- Forth, J. (2012). *Cognitively-motivated geometric methods of pattern discovery and models of similarity in music* (Doctoral dissertation). Goldsmiths, University of London.
- Forth, J., & Wiggins, G. A. (2009). An approach for identifying salient repetition in multidimensional representations of polyphonic music.
- Foubert, K., Collins, T., & De Backer, J. (2017). Impaired maintenance of interpersonal synchronization in musical improvisations of patients with borderline personality disorder. *Frontiers in psychology*, 8, 537.
- Fowler, C. B. (1966). Discovery method its relevance for music education. *Journal of Research in Music Education*, 14(2), 126–134.
- Freeman, J., Magerko, B., McKlin, T., Reilly, M., Permar, J., Summers, C., & Fruchter, E. (2014). Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch. *Proceedings of the 45th ACM technical symposium on computer science education*, 85–90.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.
- Fyfe, S., Williams, C., Mason, O. J., & Pickup, G. J. (2008). Apophenia, theory of mind and schizotypy: Perceiving meaning and intentionality in randomness. *Cortex*, 44(10), 1316–1325.
- Gabrielsson, A. (1987). Once again: The theme from mozart's piano sonata in a major. *Action and Perception in Rhythm and Music*, 81–103.
- Gamma, E. (1995). *Design patterns: Elements of reusable object-oriented software*. Pearson Education India.
- Ganguli, K. K., Lele, A., Pinjani, S., Rao, P., Srinivasamurthy, A., & Gulati, S. (2017). Melodic shape stylization for robust and efficient motif detection in hindustani vocal music. *Proceedings of the 23rd National Conference on Communications (NCC)*, 1–6.
- Garfinkle, D., Arthur, C., Schubert, P., Cumming, J., & Fujinaga, I. (2017). Pattern-finder: Content-based music retrieval with music21. *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, 5–8.
- Geist, K., Geist, E. A., & Kuznik, K. (2012). The patterns of music. *Young Children*, 2, 75.
- Giesa, T., Spivak, D. I., & Buehler, M. J. (2011). Reoccurring patterns in hierarchical protein materials and music: The power of analogies. *BioNanoScience*, 1(4), 153–161.
- Giraud, M., Groult, R., & Levé, F. (2016). Computational analysis of musical form. *Computational music analysis* (pp. 113–136). Springer.
- Gjerdingen, R. (2007). *Music in the galant style*. Oxford University Press.

- Gjerdingen, R. (2014). "Historically informed" corpus studies. *Music Perception: An Interdisciplinary Journal*, 31, 192–204.
- Glattfelder, J. B. (2019). The semantics of symmetry, invariance, and structure. *Information—consciousness—rtheeality* (pp. 65–92). Springer.
- Goetschius, P. (1904). *Lessons in music form: A manual of analysis of all the structural factors and designs employed in musical composition* (Vol. 1). Library of Alexandria.
- Goldstone, R. L., & Son, J. Y. (2012). *Similarity*. Oxford University Press.
- Goyal, A., Didolkar, A., Ke, N. R., Blundell, C., Beaudoin, P., Heess, N., Mozer, M., & Bengio, Y. (2021). Neural production systems. *arXiv preprint arXiv:2103.01937*.
- Grimm, V., Frank, K., Jeltsch, F., Brandl, R., Uchmański, J., & Wissel, C. (1996). Pattern-oriented modelling in population ecology. *Science of the Total Environment*, 183(1-2), 151–166.
- Grout, B. (2020). Music teaching redefined: The ultimate guide to teaching musical patterns.
- Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5), 1–42.
- Hahn, U., Chater, N., & Richardson, L. B. (2003). Similarity as transformation. *Cognition*, 87(1), 1–32.
- Hamanaka, M., Hirata, K., & Tojo, S. (2014). Musical structural analysis database based on gttm. *Proceedings of the 15th International Society for Music Information Retrieval (ISMIR)*.
- Hamanaka, M., Hirata, K., & Tojo, S. (2015). σ Gttm III: Learning-based time-span tree generator based on pcfg. *International Symposium on Computer Music Multidisciplinary Research*, 387–404.
- Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques*. Elsevier.
- Hand, D. J. (2020). *Dark data: Why what you don't know matters*. Princeton University Press.
- Hanninen, D. A. (2003). A theory of recontextualization in music: Analyzing phenomenal transformations of repetition. *Music Theory Spectrum*, 25(1), 59–97.
- Harkleroad, L. (2006). *The math behind the music*. Cambridge University Press.
- Hart, V. et al. (2009). Symmetry and transformations in the musical plane. *Proceedings of Bridges*, 169–176.
- Hebart, M. N., Zheng, C. Y., Pereira, F., & Baker, C. I. (2020). Revealing the multidimensional mental representations of natural objects underlying human similarity judgements. *Nature Human Behaviour*, 4(11), 1173–1185.
- Herremans, D., & Chew, E. (2017). Morpheus: Generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*.
- Honing, H., ten Cate, C., Peretz, I., & Trehub, S. E. (2015). Without it no music: Cognition, biology and evolution of musicality.

References

- Hook, J. (2007). David Lewin and the complexity of the beautiful. *Integral*, 21, 155–190.
- Hsu, J.-L., Chen, A. L., & Liu, C.-C. (1998). Efficient repeating pattern finding in music databases. *Proceedings of the 7th International Conference on Information and Knowledge Management*, 281–288.
- Huang, C.-Z. A., Koops, H. V., Newton-Rex, E., Dinculescu, M., & Cai, C. J. (2020). AI song contest: Human-AI co-creation in songwriting. *arXiv preprint arXiv:2010.05388*.
- Hudak, P. (2000). *The Haskell school of expression: Learning functional programming through multimedia*. Cambridge University Press.
- Hudak, P., Makucevich, T., Gadde, S., & Whong, B. (1996). Haskore music notation—an algebra of music. *Journal of Functional Programming*, 6(3), 465–484.
- Hudak, P., & Quick, D. (2018). *The Haskell school of music: From signals to symphonies*. Cambridge University Press.
- Hunt, S., Mitchell, T., & Nash, C. (2019). Automating algorithmic representations of musical structure using IGME: The Interactive Generative Music Environment. *Innovation in music*, 1, 1–18.
- Hunt, S., Mitchell, T., & Nash, C. (2020). Composing computer generated music, an observational study using IGME: the Interactive Generative Music Environment. *Proceedings of 20th International Conference on New Interfaces for Musical Expression (NIME)*.
- Hunt, S. (2020). An analysis of repetition in video game music. *Proceedings of the 1st Joint Conference on AI Music Creativity*, 7.
- Huron, D. B. (2006). *Sweet anticipation: Music and the psychology of expectation*. MIT press.
- Hutton, G. (2016). *Programming in Haskell* (2nd). Cambridge University Press.
- Janin, D. (2016). A robust algebraic framework for high-level music writing and programming. *Technologies for Music Notation and Representation (TENOR)*.
- Janssen, B. (2018). *Retained or lost in transmission?* (Doctoral dissertation). University of Amsterdam.
- Janssen, B., de Haas, W. B., Volk, A., & van Kranenburg, P. (2014). Finding repeated patterns in music: State of knowledge, challenges, perspectives. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8905, 277–297.
- Janssen, B., Van Kranenburg, P., & Volk, A. (2017). Finding occurrences of melodic segments in folk songs employing symbolic similarity measures. *Journal of New Music Research*, 46(2), 118–134.
- Jiménez, A., Molina-Solana, M., Berzal, F., & Fajardo, W. (2011). Mining transposed motifs in music. *Journal of Intelligent Information Systems*, 36(1), 99–115.
- Jindal, N., & Liu, B. (2006). Identifying comparative sentences in text documents. *Proceedings of the 29th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, 244–251.
- Jones, M. R. (1987). Dynamic pattern structure in music: Recent theory and research. *Perception & Psychophysics*, 41(6), 621–634.

- Juba, B., Kalai, A. T., Khanna, S., & Sudan, M. (2011). Compression without a common prior: An information-theoretic justification for ambiguity in language. *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS)*, 79–84.
- Kaduri, Y. (2006). Wittgenstein and Haydn on understanding music. *Contemporary Aesthetics*, 4(1), 15.
- Kahneman, D. (2011). *Thinking, fast and slow*. Macmillan.
- Kaplan, F. (2015). A map for big data research in digital humanities. *Frontiers in Digital Humanities*, 2, 1.
- Katsiavalos, A., Collins, T., & Battey, B. (2019). An initial computational model for musical schemata theory. *Proceedings of the 20th International Society for Music Information Retrieval (ISMIR)*, 166–172.
- Kempf, D. (1996). What is symmetry in music? *International Review of the Aesthetics and Sociology of Music*, 155–165.
- Kephart, R. (2017). Gregorian Chant Notation.
- Klaus Frieler, M. P., Frank Höger, & Dixon, S. (2018). Two web applications for exploring melodic patterns in jazz solos. *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 777–783.
- Knopke, I., & Jürgensen, F. (2009). A system for identifying common melodic phrases in the masses of palestrina. *Journal of New Music Research*, 38(2), 171–181.
- Kohonen, T. (1990). Improved versions of learning vector quantization. *International Joint Conference on Neural Networks (IJCNN)*, 545–550.
- Koops, H. V., de Haas, W. B., Bountouridis, D., & Volk, A. (2016). Integration and quality assessment of heterogeneous chord sequences using data fusion. *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 178–184.
- Koops, H. V., de Haas, W. B., Burgoyne, J. A., Bransen, J., Kent-Muller, A., & Volk, A. (2019). Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48(3), 232–252.
- Koops, H. V., Magalhães, J. P., & de Haas, W. B. (2013). A functional approach to automatic melody harmonisation. *Proceedings of the 1st ACM SIGPLAN Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 47–58.
- Kortylewski, A., He, J., Liu, Q., Cosgrove, C., Yang, C., & Yuille, A. L. (2021). Compositional generative networks and robustness to perceptible image changes. *Proceedings of 55th Annual Conference on Information Sciences and Systems (CISS)*, 1–8.
- Krumhansl, C. L., Sandell, G. J., & Sergeant, D. C. (1987). The perception of tone hierarchies and mirror forms in twelve-tone serial music. *Music Perception: An Interdisciplinary Journal*, 5(1), 31–77.
- Kubik, G. (1979). Pattern perception and recognition in African music. *The Performing Arts: Music and Dance*, 221–49.
- Kursa, M. B., Rudnicki, W. R. et al. (2010). Feature selection with the Boruta package. *J Stat Software*, 36(11), 1–13.

References

- Lartillot, O. (2004). A musical pattern discovery system founded on a modeling of listening strategies. *Computer Music Journal*, 28(3), 53–67.
- Lartillot, O. (2005). Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 229–235.
- Lartillot, O. (2011). MIRtoolbox 1.3. 4 user's manual. *Finnish Centre of Excellence in Interdisciplinary Music Research, University of Jyväskylä, Finland*.
- Lartillot, O. (2014). PatMinr: In-depth motivic analysis of symbolic monophonic sequences.
- LaRue, J. (1992). *Guidelines for style analysis*. Harmonie Park Press.
- Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9), 1060–1076.
- Lent, J. (2017). *The patterning instinct*. Prometheus Books, Amherst New York.
- Lerdahl, F., & Jackendoff, R. (1983). An overview of hierarchical structure in music. *Music Perception*, 229–252.
- Lerdahl, F., & Jackendoff, R. S. (1985). *A generative theory of tonal music*. MIT press.
- Lewin, D. (1979). A response to a response: On pcset relatedness. *Perspectives of New Music*, 498–502.
- Lewin, D. (1987). *Generalized musical intervals and transformations*. Oxford University Press, USA.
- Lewin, D. (2008). Transformational considerations in Schoenberg's opus 23, number 3. *Music theory and mathematics: chords, collections, and transformations*, 197–221.
- Lidov, D. (2005). *Is language a music?: Writings on musical form and signification*. Indiana University Press.
- Lin, C.-R., Liu, N.-H., Wu, Y.-H., & Chen, A. L. (2004). Music classification using significant repeating patterns. *Proceedings of the International Conference on Database Systems for Advanced Applications*, 506–518.
- Lipovaca, M. (2011). *Learn you a Haskell for great good!: A beginner's guide*. No Starch Press.
- Liu, X. F., Chi, K. T., & Small, M. (2010). Complex network structure of musical compositions: Algorithmic generation of appealing music. *Physica A: Statistical Mechanics and its Applications*, 389(1), 126–132.
- Lockhart Nelson, S. (2017). Pattern and meaning. *Interalia Magazine*.
- London, J. (2021). Music theory as junk science, and how and why we need to fix it.
- Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 431–439.
- Lovelace, A. (1843). Sketch of the analytical engine invented by charles babbage, esq. with notes by the translator. *Scientific Memoirs*, 3, 666–731.
- Macdonald, H. (2001). Transformation, thematic.
- Macdonald, J., Wäldchen, S., Hauch, S., & Kutyniok, G. (2019). A rate-distortion framework for explaining neural network decisions. *arXiv preprint arXiv:1905.11092*.

- Magalhães, J. P., & Koops, H. V. (2014). Functional generation of harmony and melody. *Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 11–21.
- Mandelbrot, B., & Hudson, R. L. (2007). *The misbehavior of markets: A fractal view of financial turbulence*. Basic books.
- Mannone, M. (2018). Introduction to gestural similarity in music: An application of category theory to the orchestra. *Journal of Mathematics and Music*, 12(2), 63–87.
- Mansouri, F. A., Freedman, D. J., & Buckley, M. J. (2020). Emergence of abstract rules in the primate brain. *Nature Reviews Neuroscience*, 21(11), 595–610.
- Margolis, H. (1987). *Patterns, thinking, and cognition: A theory of judgment*. University of Chicago Press.
- Margulis, E. H. (2014). *On repeat: How music plays the mind*. Oxford University Press.
- Mazzola, G. (2018). *The topos of music I: Theory: Geometric logic, classification, harmony, counterpoint, motives, rhythm*. Springer.
- Mazzola, G., Mannone, M., & Pang, Y. (2016). *Cool math for hot music*. Springer.
- McAdams, S., & Matzkin, D. (2001). Similarity, invariance, and musical variation. *Annals of the New York Academy of Sciences*, 930(1), 62–76.
- McKay, C. (2010). *Automatic music classification with jMIR* (Doctoral dissertation). McGill University.
- McLean, A. (2014). Making programming languages to dance to: Live coding with tidal. *Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling & Design (FARM)*, 63–70.
- McLean, A., & Harlizius-Klück, E. (2018). Fabricating algorithmic art.
- McLean, A., & Wiggins, G. (2010). Tidal-pattern language for the live coding of music. *Proceedings of the 7th sound and music computing conference (SMC)*.
- McLeod, A., Owers, J., & Yoshii, K. (2020). The midi degradation toolkit: Symbolic music augmentation and correction. *arXiv preprint arXiv:2010.00059*.
- Melkonian, O. (2019). Music as language: Putting probabilistic temporal graph grammars to good use. *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 1–10.
- Melkonian, O., Ren, I. Y., Swierstra, W., & Volk, A. (2019). What constitutes a musical pattern? *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 95–105.
- Meredith, D. (2006). The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35(2), 121–159.
- Meredith, D. (2013). COSIATEC and SIATECCOMPRESS: Pattern discovery by geometric compression.
- Meredith, D. (2015). Music analysis and point-set compression. *Journal of New Music Research*, 44(3), 245–270.
- Meredith, D. (2019). RECURSIA-RRT: Recursive translatable point-set pattern discovery with removal of redundant translators. *Proceedings of the 12th International Workshop on Machine Learning and Music (MML)*.

References

- Meredith, D., Lemström, K., & Wiggins, G. A. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4), 321–345.
- Meredith, D., Wiggins, G. A., & Lemström, K. (2001). Pattern induction and matching in polyphonic music and other multidimensional datasets. *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics*, 22–25.
- Meredith, M. (1991). Data modeling: A process for pattern induction. *Journal of Experimental & Theoretical Artificial Intelligence*, 3(1), 43–68.
- Meyer, L. B. (1973). *Explaining music: Essays and explorations*. University of California Press.
- Meyer, M. F. (1929). *The musician's arithmetic: Drill problems for an introduction to the scientific study of musical composition* (Vol. 4). Columbia: The University of Missouri.
- Meyer-Vitali, A., Bakker, R., van Bekkum, M., de Boer, M., Burghouts, G., van Digenlen, J., Dijk, J., Grappiolo, C., de Greeff, J., Huizing, A., et al. (2019). Hybrid ai: White paper. *TNO Reports*.
- Middleton, R. (1990). *Studying popular music*. McGraw-Hill Education.
- Mikolov, T., Yih, W.-t., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 746–751.
- Milewski, B., & Tabachnik, I. (2018). *Category theory for programmers*. Blurb, Incorporated.
- Mitchell, H. B. (2012). *Data fusion: Concepts and ideas*. Springer Science & Business Media.
- Mithen, S. (1996). *The prehistory of the mind: The cognitive origins of art and science*. Thames & Hudson Ltd.
- Monelle, R. (2014). *Linguistics and semiotics in music*. Routledge.
- Müllensiefen, D. (2009). Fantastic: Feature analysis technology accessing statistics (in a corpus). *Goldsmiths University of London*.
- Müllensiefen, D., Gingras, B., Musil, J., & Stewart, L. (2014). The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PLoS one*, 9(2), e89642.
- Müller, M., & Jiang, N. (2012). A scape plot representation for visualizing repetitive structures of music recordings. *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR)*, 97–102.
- Mumford, D. (1994). Pattern theory: A unifying perspective. *Proceedings of the 1st European congress of mathematics*, 187–224.
- Nettl, B. (1956). Unifying factors in folk and primitive music. *Journal of the American Musicological Society*, 9(3), 196–201.
- Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 841–848.

- Nieto, O., & Farbood, M. M. (2012a). Perceptual evaluation of automatically extracted musical motives. *Proceedings of the 12th International Conference on Music Perception and Cognition (ICMPC)*, 723–727.
- Nieto, O. (2015). *Discovering structure in music: Automatic approaches and perceptual evaluations* (Doctoral dissertation). New York University.
- Nieto, O., & Farbood, M. M. (2014). Identifying polyphonic patterns from audio recordings using music segmentation techniques. *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 411–416.
- Nieto, O., & Farbood, M. M. (2012b). Perceptual evaluation of automatically extracted musical motives. *Proceedings of the 12th International Conference on Music Perception and Cognition, ICMPC*, 723–727.
- Nilsson, H., & Chupin, G. (2017). Funky grooves: Declarative programming of full-fledged musical applications. *International Symposium on Practical Aspects of Declarative Languages*, 163–172.
- Noether, E. (1918). Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen. Mathematisch-Physikalische Klasse*. pp235-257.
- Northcutt, C., Jiang, L., & Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70, 1373–1411.
- Northcutt, C. G., Athalye, A., & Mueller, J. (2021). Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*.
- Ockelford, A. (2017). *Repetition in music: Theoretical and metatheoretical perspectives*. Routledge.
- Ockelford, A. (2018). *Comparing notes*. Simon; Schuster.
- Oxford, D. (2021). Pattern, n. and adj. : Oxford english dictionary. packages by category | Hackage. (2020).
- Papadimitriou, S., Sun, J., & Faloutsos, C. (2005). Streaming pattern discovery in multiple time-series. *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, 697–708.
- Parida, L. (2007). *Pattern discovery in bioinformatics: Theory & algorithms*. Chapman; Hall/CRC.
- Park, S., Kwon, T., Lee, J., Kim, J., & Nam, J. (2019). A cross-scape plot representation for visualizing symbolic melodic similarity. *Proceedings of the 20th International Society for Music Information Retrieval (ISMIR)*, 423–430.
- Patel, A. D. (2003). Language, music, syntax and the brain. *Nature neuroscience*, 6(7), 674–681.
- Patel, S., & Barkovich, A. J. (2002). Analysis and classification of cerebellar malformations. *American Journal of Neuroradiology*, 23(7), 1074–1087.
- Pearce, M. T., & Wiggins, G. A. (2007). Evaluating cognitive models of musical composition. *Proceedings of the 4th international joint workshop on computational creativity*, 73–80.
- Pearce, M. T. (2005). *The construction and evaluation of statistical models of melodic structure in music perception and composition* (Doctoral dissertation). City University London.

References

- Pease, A., Colton, S., Warburton, C., Nathanail, A., Preda, I., Arnold, D., Winterstein, D., & Cook, M. (2019). The importance of applying computational creativity to scientific and mathematical domains. *Proceedings of the 10th International Conference on Computational Creativity (ICCC)*, 250–257.
- Pesek, M., Tomašević, D., Ren, I. Y., & Marolt, M. An opensource web-based pattern annotation framework - PAF. In: *Late-breaking Demos of ISMIR*. 2019, 1–2.
- Pesek, M., Leonardis, A., & Marolt, M. (2017). SymCHM—an unsupervised approach for pattern discovery in symbolic music with a compositional hierarchical model. *Applied Sciences*, 7(11), 1135.
- Petricek, T. (2019). Cultures of programming.
- Pfleiderer, M., Frieler, K., & Abersser, J. (2019). melpat — The Jazzomat research project.
- Phrase. (2001).
- Pickering, M., Gibbons, J., & Wu, N. (2017). Profunctor optics: Modular data accessors. *Art, Science, and Engineering of Programming*, 1(2), 7.
- Popoff, A. (2012). Towards a categorical approach of transformational music theory. *arXiv preprint arXiv:1204.3216*.
- Quick, D. (2018). Demo: Pattern-based algorithmic music with euterpea. *Proceedings of the 6th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 38–39.
- Quick, D., & Thomas, K. (2019). A functional model of jazz improvisation. *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 11–21.
- Rahn, J. (1993). Repetition. *Contemporary Music Review*, 7(2), 49–57.
- Rahn, J. (2004). The swerve and the flow: Music's relationship to mathematics. *Perspectives of New Music*, 42(1), 130–148.
- Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the 43th International Conference on Very Large Data Bases (VLDB)*, 11(3), 269.
- Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., & Ré, C. (2017). Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 3236–3246.
- Reminiscence motif. (2002).
- Ren, I., Volk, A., Swierstra, W., & Veltkamp, R. C. (2020). A computational evaluation of musical pattern discovery algorithms.
- Ren, I. Y. (2016). Closed patterns in folk music and other genres. *Proceedings of the 6th International Workshop on Folk Music Analysis (FMA)*, 56–58.
- Ren, I. Y., Koops, H. V., Volk, A., & Swierstra, W. (2017). In search of the consensus among musical pattern discovery algorithms. *Proceedings of the 18th International Society for Music Information Retrieval (ISMIR)*, 671–680.
- Ren, I. Y., Koops, H. V., Volk, A., & Swierstra, W. (2018). Investigating musical pattern ambiguity in a human annotated dataset. *Proceedings of the 15th International Conference on Music Perception and Cognition and the 10th triennial*

- conference of the European Society for the Cognitive Sciences of Music (ICM-PC/ESCOM), 361–367.
- Ren, I. Y., Volk, A., Swierstra, W., & Veltkamp, R. C. (2018). Analysis by classification: A comparative study of annotated and algorithmically extracted patterns in symbolic music data. *Proceedings of the 19th International Society for Music Information Retrieval (ISMIR)*, 539–546.
- Research, D. S. (2019). Towards robust and verified AI: specification testing, robust training, and formal verification. *Medium*.
- Reti, R. (1951). *The thematic process in music*. Macmillan.
- Rings, S. (2011). *Tonality and transformation*. Oxford University Press.
- Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge University Press.
- Robertson, R., & Combs, A. (1999). *Chaos theory in psychology and the life sciences*. Amsco Publications.
- Rodríguez López, M. (2016). *Automatic melody segmentation* (Doctoral dissertation). Utrecht University.
- Rodríguez López, M. E., Volk, A., & Bountouridis, D. (2014). Multi-strategy segmentation of melodies. *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR)*, 207–212.
- Rolland, P.-Y. (1998). *Découverte automatique de regularités dans les séquences et application à l'analyse musicale* (Doctoral dissertation). Paris 6.
- Rolland, P.-Y. (1999). Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4), 334–350.
- Romanycia, M. H., & Pelletier, F. J. (1985). What is a heuristic? *Computational Intelligence*, 1(1), 47–58.
- Roth, D. (2017). Incidental supervision: Moving beyond supervised learning. *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 31(1).
- Russell, S. (2019). *Human compatible: Artificial intelligence and the problem of control*. Penguin.
- Ruthmann, A., Heines, J., Greher, G., Laidler, P., & II, C. (2010). Teaching computational thinking through musical live coding in scratch. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE)*, 351–355.
- Saxena, K., & Rajpoot, D. (2009). A way to understand various patterns of data mining techniques for selected domains. *arXiv preprint arXiv:0911.0781*.
- Scerri, E. (2019). *An approach for automated pattern discovery in symbolic music with long short-term memory neural networks* (Master's thesis). Utrecht University.
- Schafer, R. W. (2011). What is a Savitzky-Golay filter? *IEEE Signal Processing Magazine*, 28(4), 111–117.
- Scheel, A. M., Tiokhin, L., Isager, P. M., & Lakens, D. (2020). Why hypothesis testers should spend less time testing hypotheses. *Perspectives on Psychological Science*, 1745691620966795.
- Schenker, H. (1980). *Harmony*. University of Chicago Press.

References

- Scherder, E. (2017). *Singing in the brain: Over de unieke samenwerking tussen muziek en de hersenen*. Singel Uitgeverijen.
- Schnapper, L. (2001). Ostinato.
- Schoenberg, A. (1967). *Fundamentals of musical composition*. Faber; Faber.
- Schoenberg, A. (2006). *The musical idea and the logic, technique and art of its presentation*. Indiana University Press.
- Schubert, P. N. (1995). A lesson from lassus: Form in the duos of 1577. *Music theory spectrum*, 17(1), 1–26.
- Sears, D. R., & Widmer, G. (2020). Beneath (or beyond) the surface: Discovering voice-leading patterns with skip-grams. *Journal of Mathematics and Music*, 1–26.
- Serafine, M. L. (1988). *Music as cognition: The development of thought in sound*. Columbia University Press.
- Shams, Z., Jamnik, M., Stapleton, G., & Sato, Y. (2018). Icon: A diagrammatic theorem prover for ontologies. *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 204–208.
- Shermer, M. (2008). Patternicity: Finding meaningful patterns in meaningless noise. *Scientific American*, 299(6), 48.
- Silva, N., Fischione, C., & Turchet, L. (2020). Towards real-time detection of symbolic musical patterns: Probabilistic vs. deterministic methods. *Proceedings of the 27th Conference of Open Innovations Association (FRUCT)*, 238–246.
- Simard, P., LeCun, Y., & Denker, J. S. (1993). Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems*, 50–58.
- Simard, P. Y., LeCun, Y. A., Denker, J. S., & Victorri, B. (1998). Transformation invariance in pattern recognition—tangent distance and tangent propagation. *Neural networks: Tricks of the trade* (pp. 239–274). Springer.
- Simon, H. A., & Sumner, R. K. (1993). Pattern in music. *Machine models of music*, 83–110.
- Simonton, D. K. (2013). Creative thought as blind variation and selective retention: Why creativity is inversely related to sightedness. *Journal of Theoretical and Philosophical Psychology*, 33(4), 253.
- Slonimsky, N. (1999). *Thesaurus of scales and melodic patterns*. Amsco Publications.
- Snyder, B., & Snyder, R. (2000). *Music and memory: An introduction*. MIT press.
- Spiegel, L. (1981). Manipulations of musical patterns. *Proceedings of the Symposium on Small Computers and the Arts*, 19–22.
- Stech, D. A. (1981). A computer-assisted approach to micro-analysis of melodic lines. *Computers and the Humanities*, 15(4), 211–221.
- Stegemann, T., Geretsegger, M., Phan Quoc, E., Riedl, H., & Smetana, M. (2019). Music therapy and other music-based interventions in pediatric health care: An overview. *Medicines*, 6(1), 25.
- Steyerl, H. (2016). A sea of data: Apophenia and pattern (mis-) recognition. *E-flux Journal*, 72.

- Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3), 293–300.
- Szamozvancev, D., & Gale, M. B. (2017). Well-typed music does not sound wrong (experience report). *Proceedings of the 10th ACM SIGPLAN International Symposium on Haskell*, 99–104.
- Temperley, D. (1995). Motivic perception and modularity. *Music Perception: An Interdisciplinary Journal*, 13(2), 141–169.
- Temperley, D. (2014). Information flow and repetition in music. *Journal of Music Theory*, 58(2), 155–178.
- Tenney, J., & Polansky, L. (1980). Temporal gestalt perception in music. *Journal of Music Theory*, 24(2), 205–241.
- The AI song contest. (2020).
- Theme (jazz). (2003).
- Thorpe, M., Ockelford, A., & Aksentijevic, A. (2012). An empirical exploration of the zygonic model of expectation in music. *Psychology of Music*, 40(4), 429–470.
- Toiviainen, P., Luck, G., & Thompson, M. R. (2010). Embodied meter: Hierarchical eigenmodes in music-induced movement. *Music Perception*, 28(1), 59–70.
- Topham, T. (2020). Music teaching redefined: The ultimate guide to left hand piano styles patterns.
- Toussaint, E. R., Toussaint, G. T. et al. (2014). What is a pattern. *Proceedings of Bridges*, 293–300.
- Tukey, J. W., & Wilk, M. B. (1966). Data analysis and statistics: An expository overview. *Proceedings of the Fall Joint Computer Conference*, 695–709.
- Turner, R., & Eden, A. H. (2008). The philosophy of computer science. *Journal of Applied Logic*, 6(4), 459.
- Tymoczko, D. (2009). Generalizing musical intervals. *Journal of Music Theory*, 53(2), 227–254.
- Typke, R., Veltkamp, R. C., & Wiering, F. (2006). A measure for evaluating retrieval techniques based on partially ordered ground truth lists. *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 1793–1796.
- Typke, R., Wiering, F., & Veltkamp, R. C. (2005). A survey of music information retrieval systems. *Proceedings of the 6th international Society on Music Information Retrieval (ISMIR)*, 153–160.
- Typke, R., Wiering, F., & Veltkamp, R. C. (2007). Transportation distances and human perception of melodic similarity. *Musicae Scientiae*, 11(1), 153–181.
- Utgoff, P. (2006). Detecting motives and recurring patterns in polyphonic music. *Proceedings of the 30th International Computer Music Conference (ICMC)*, 487–494.
- Valdesolo, P., & Graham, J. (2014). Awe, uncertainty, and agency detection. *Psychological science*, 25(1), 170–178.
- Van Balen, J. (2016). *Audio description and corpus analysis of popular music* (Doctoral dissertation). University Utrecht.

References

- van Kranenburg, P., Janssen, B., & Volk, A. (2016). The Meertens Tune Collections: The Annotated Corpus (MTC-ANN) versions 1.1 and 2.0.1. *Meertens Online Reports*, (1).
- van Kranenburg, P., Volk, A., & Wiering, F. (2013). A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1), 1–18.
- Velarde, G., Meredith, D., & Weyde, T. (2016). A wavelet-based approach to pattern discovery in melodies. *Computational music analysis* (pp. 303–333). Springer.
- Velarde, G., Weyde, T., & Meredith, D. (2013). An approach to melodic segmentation and classification based on filtering with the haar-wavelet. *Journal of New Music Research*, 42(4), 325–345.
- Vieira, R., & Schiavoni, F. L. (2020). Can pipe-and-filters architecture help creativity in music? *Proceedings of the Workshop de Música Ubiqua (UbiMus)*, 109.
- Volk, A., de Haas, W. B., & van Kranenburg, P. (2012). Towards modelling variation in music as foundation for similarity. *Proceedings of the 12th International Conference on Music Perception and Cognition ICMPC*, 1085–1094.
- Volk, A., & Van Kranenburg, P. (2012). Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *MusicæScientiæ*, 16(3), 317–339.
- Von Neumann, J. *et al.* (1958). Computer and the brain.
- Vreeken, J., Van Leeuwen, M., & Siebes, A. (2011). Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1), 169–214.
- Wang, J. T.-L., Chirn, G.-W., Marr, T. G., Shapiro, B., Shasha, D., & Zhang, K. (1994). Combinatorial pattern discovery for scientific data: Some preliminary results. *Proceedings of the International Conference on Management of Data ACM SIGMOD*, 115–125.
- Webern, A. (1963). *Cited in boulez, music and philosophy, edward cambell*. Cambridge University Press.
- Wells, S. (2019). *Creating a tool for facilitating and researching human annotation of musical patterns* (Master's thesis). Utrecht University.
- Wertheimer, M. (1938). Gestalt theory.
- Weyl, H. (2015). *Symmetry* (Vol. 104). Princeton University Press.
- Whitehead, A. N., & Price, L. (2001). *Dialogues of alfred north whitehead* (Vol. 84). David R. Godine Publisher.
- Whittall, A. (2001). Leitmotif.
- Wiering, F., de Nooijer, J., Volk, A., & Tabachneck-Schijf, H. J. (2009). Cognition-based segmentation for music information retrieval systems. *Journal of New Music Research*, 38(2), 139–154.
- Wiggins, G. A. (1998). Music, syntax, and the meaning of “meaning”. *Proceedings of the 1st Symposium on Music and Computers*, 18–23.
- Wiggins, G. A. (2006). Searching for computational creativity. *New Generation Computing*, 24(3), 209–222.
- Witmer, R. (2001). Lick.

- Wu, S.-L., & Yang, Y.-H. (2020). The jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. *arXiv preprint arXiv:2008.01307*.
- Yang, C., Shen, Y., & Zhou, B. (2019). Semantic hierarchy emerges in deep generative representations for scene synthesis. *arXiv preprint arXiv:1911.09267*.
- Yi, H. (2013). *On symmetry: A framework for automated symmetry detection* (Doctoral dissertation). University of Maryland.
- Young, H. (2017). A categorial grammar for music and its use in automatic melody generation. *Proceedings of the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM)*, 1–9.
- Zalkow, F., Balke, S., Arifi-Müller, V., & Müller, M. (2020). MTD: A multimodal dataset of musical themes for MIR research. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1).
- Zalta, E. N., Nodelman, U., Allen, C., & Anderson, R. L. (2005). Stanford encyclopedia of philosophy. *Stanford University*.
- Zbikowski, L. (2002). *Conceptualizing music: Cognitive structure, theory, and analysis*. Oxford University Press on Demand.
- Zhu, Y., Gharghabi, S., Silva, D. F., Dau, H. A., Yeh, C.-C. M., Senobari, N. S., Almaslukh, A., Kamgar, K., Zimmerman, Z., Funning, G., et al. (2020). The swiss army knife of time series data mining: Ten useful things you can do with the matrix profile and ten lines of code. *Data Mining and Knowledge Discovery*, 34(4), 949–979.