FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Automated Music Generation
Student:	Adam Beckert
Supervisor:	doc. RNDr. Pavel Surynek, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2021/2022

Instructions

The aim of this thesis is the automated generation of music composition. Since the task of music generation is very complex, the thesis will focus on specific sub-problems such as the connection of two songs together. The research is expected to be done at the symbolic level using the commonly accepted MIDI format. The student will fulfill following tasks:

1. Study relevant literature on music theory and music generation at the symbolic level; prepare a survey; identify important sub-problems.

2. Adapt an existing concept from computer science and apply it on selected sub-problem from music generation; the candidate concept are for example formal grammars.
3. Implement the proposed method as a software prototype and perform relevant tests focused on the quality of generated music.

–

[1] Valentin Emiya, Roland Badeau, Bertrand David: Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. Proceedings of the 16th European Signal Processing Conference (EUSIPCO 2008), pp. 1-5, IEEE Computer Society, 2008.

[2] Nierhaus, Gerhard: Algorithmic Composition, Paradigms of Automated Music Generation. Springer Verlag, 2019.

[3] Donya Quick, Paul Hudak: Grammar-based automated music composition in Haskell. FARM '13: Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design, ACM, 2013.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 30 September 2020 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Automated music generation

Adam Beckert

Department of Theoretical Computer Science
Supervisor: doc. RNDr. Pavel Surynek, Ph.D.

June 27, 2021

Declaration

In Prague on June 27, 2021 . . . .

Citation of this thesis

Beckert, Adam. *Automated music generation.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Tato bakalářská práce se zabývá algoritmickým generováním hudby. Práce shrnuje aktuální řešení automatické kompozice a jejich výsledky. Jsou zde převážně zmíněné rozdíly mezi jednotlivými přístupy a jejich nevýhodami. V další části jsou popsány prvky z hudební teorie, teorie jazyků a základy pravděpodobnosti. Ty jsou dále využity k vytvoření systému ke generování přechodu mezi dvěma skladbamy.

Praktická část definuje omezení na skladby použité pro generování. Skladby jsou definované ve formátu MIDI. Dále je práce rozdělena do dvou částí: analýzy skladeb a procesu generování. U analýzy je použito znalosti z teorie hudby pro reprezentaci a analýzy skladeb. Následně s použitím Markovových řetězců je vytvořen model, který slouží ke generování nové skladby.

Testování je následně provedeno se čtyřmi písní a vzniklé skladby jsou posouzeny dobrovolníkem za pomoci předem definovaných charakteristik.

Klíčová slova hudba, teorie hudby, generování hudby, stavové systémy, Scala, Markovovy řetězce

Abstract

This bachelor thesis describes the algorithmic generation of music, it gives an overview of existing solutions in the process of automatic composition and their results. This summary predominantly describes the advantages and disadvantages of different approaches to the issue.

The next section describes structures from music theory, state transition systems, and probabilistic introduction. These structures are the basis for the synthesis of the transition between two songs.

The practical part defines constraints on the MIDI songs used as input to the system. The system is separated into two parts: the analyzer and the generator. The analysis uses concepts from music theory to create an internal representation, which is then used to create Markov chains. This model and other attributes from the analysis are then given to the generator.

The system is subjectively evaluated.

Keywords music, music theory, automatic music generation, algorithmic composition, state transition system, Scala, Markov chains, algorithmic composition

Contents

List of Figures

List of Tables

Introduction

Ever since the creation of computers, there were attempts of algorithmic composition and use of *computational creativity*. With recent machine learning techniques, computational creativity is mainly associated with automated analysis of songs and synthesis of new combinations similar to the data used. Unlike other machine learning techniques, such as object detection, music is not easily evaluated algorithmically, so the approach of subjective validation is selected.

This thesis focuses on the composition of a short transition that connects two songs. In the process of *algorithmic composition*, several steps are created from analyzing existing songs and synthesis of generic songs to the final composition of the transition.

A summary of different types of state-of-the-art solutions of algorithmic composition is created. Based on that, an approach to solve the issue is selected. The system is composed largely of Markov chains in the process of analysis, on the other hand, the generator uses probabilistic context-free grammars.

Motivation

Solving automated music generation is in high demand as the music industry revenue is in dozens of billion USD yearly. The most obvious application is creating new songs for performers or even creating new personalized songs. Moreover, more recent applications are in video games and movie soundtracks, where the musical piece evokes emotions and brings a more vivid experience to the customers.

The obstacle in automatic music generation is that this topic spans over several fields – computer science, musicology, math, and physics. Already Musicology is based on ethnology, history, psychology, acoustics, mathematics, philosophy, and art. This intersection of fields often creates miscommunication between technical and artistic views and inconsistencies in scientific publications. Luckily, several overviews of existing studies on the topic of automatic music composition exist. The overview ”A Functional Taxonomy of Music Generation Systems”[1] and ”AI Methods in Algorithmic Composition: A Comprehensive Survey”[2] were used as a source for state-of-the-art knowledge on the topic of automatic composition.

In the artistic view, creativity describes one’s character and expresses emotions in the art. On the other hand, a technical representation of music is an attempt to formalize structures found in western music. Music theory is oriented mainly toward describing western music in a more formal representation. This thesis will cover the fundamentals of music theory, and when referred to music, it is related to western music, specifically tonal music. Throughout history, patterns like harmony and tonality emerged, adding new features to separate previous works from newer ones. The most studied genres in automatic music generation are classical music, jazz, and pop genre.

Structure of the thesis

The thesis is separated into 5 parts. The first is the background and an *overview of the state-of-the-art* solutions in algorithmic composition. The listed approaches are described in a quick summary, with some examples of the existing work. The chapter representation of music lists different types of notation used to describe music and most of them will be used in this work. *MIDI standard* is described in more detail as it is used as input to the system. The individual signals of MIDI are listed and a few downsides of this representation are described.

In the chapter Music theory, different concepts from the theory of music are described. The transition from sound to individual notes, relationships between individual notes, and further development of these relationships to more complex structures is shown. At the end of the chapter, a quick overview of the listed structures and connection with the qualities of tonal music is established.

In chapter Computational background in music, the event space is defined and later followed by the probability. After that, an overview of the possible event spaces in terms of music is described. Next is the extraction of some attributes from the song with the help of *correlation*, which is used for identifying the scale and chord progressions. In the last part, *state transition systems, stochastic context-free grammars, and Markov chains* are introduced.

The practical part consists of 5 main sections. The first is the introduction of the *constraints on the song* and the qualities that were defined. The *preprocessing* part deals with constructing internal representation from the MIDI format of the input songs. Some of the issues related to the MIDI format are described. The system – Electronic Virutal Artistist (EVA), consists of two parts. One is the *analyzer* and second *generator*. In the analyzer, the song is represented in structures of music theory, and attributes are extracted from it. From the internal representation, Markov chains are trained, and together with the attributes are used as input to the generator. The generator establishes a fixed format for synthesis of the transition and for the evaluation of the system,

Chapter 1

Related Work

The topic of automatic music generation has been a long-studied problem, and this issue is still in development. The first computer-generated compositions are dated to 1957 [3] making this topic almost as old as the computers themselves. Even before the first computers were invented, there were tendencies by several composers to automate music generation, such as Mozart's Würfelspiel (Dice games [1]) The range of approaches to algorithmic composition is very wide, enclosing methods from machine learning, musicology, mathematical fields. . . . The next sections cover only a small part of the most used methods. The methods are grouped into sections based on their internal structure. In the process of algorithmic composition, methods are typically combined.

From a computer science point of view, we have several models that are used for generation. The main ones are *Grammars*, *Knowledge and rule-based systems* (KBS), *Markov chains* (MM), *Artificial neural networks* (ANN), *Evolutionary algorithms* (EA),*Cellular automata* (CA). In recent years, the success of artificial neural networks has led to applying this method to various fields. The research is currently saturated with this type of model, in spite of that, I would like to put emphasis on knowledge-based systems, probabilistic grammars, and Markov chains, as these are the models used in this work.

1.1 Knowledge and rule-based systems

Systems, where knowledge is used to create structured symbols (or a set of rules) and dependencies between the symbols. Typically KBS is used in *conjunction with other approaches* as some level of musical composition is known,

[1]These games were developed in the 18th century. They were based on many several variations of parts of the song. Dice throw was used as a random selection from this set of variations, consequently, creating a unique set of combinations, thus a newly composed song. The earliest work of dice games is expected to be from J.P. Kirnberger [4]

but some aspects remain unclear. *The theory of music* and its sub-theories represent music in musical symbols. The notation is one of the examples, where the key features of the songs are captured. Even though the rules are defined statically, they can also be used to learn probabilities for some of the structures. Typically, machine learning techniques for chord progressions or melody are used.

KBS can be created from systems of harmonization, counterpoint, jazz improvisation rules, theories such as *"A Generative Theory of Tonal Music"* [5] or use of various spaces describing dependencies between structures(chordal spaces, pitch spaces).

A generative theory of tonal music is a theory created by American composers Fred Lerdahl and Ray Jackendoff, with an attempt to formalize music in the style of *Noam Chomsky's grammar.* First published in 1983, it was created to define musical grammar to describe how people understand music scientifically. As both authors of this theory are experts in their field and have dozens of books published on the algorithmic composition of music and representation of music structures, GTTM is widely cited and influenced many researchers.

GTTM creates a hierarchical system that is tightly connected to musical intuition. The structures defined: *Grouping structure, Metrical structure, Time-span reduction, Prolongation reduction.* Each of these structures has a set of rules that define how the structures are formed, and they describe how to form the internal representation as a *binary tree.* GTTM is a quite extensive theory, and so far, nobody has managed to create a system that would cover all the rules. Usually, studies limit the set of rules and focus only on some structures. (etc. "Musical structural analysis database based on GTTM" [6] focuses on Time-span reduction) GTTM is also used for the analysis of musical pieces, in a study by Masatoshi Hamanaka [7], 17 out of 26 rules were implemented.

1.2 Grammars

Grammars can be described as a set of production rules which expand *high-level symbols* into *sequences of terminal symbols.* The sequences (words) are created by repeatedly applying rules, this is why they are suited for the representation of knowledge-based systems. A set of all sequences that are created by derivation rules from the grammar form a language.

The grammars can be created from the author's knowledge of music, but they can be derived from analysis of the corpus. Many types of grammar can be used in algorithmic composition. Amongst the most used are *stochastic context-free grammars.* These grammars turned to have quite promising results for some studied genres such as chorales, etudes, and jazz.[8, 9] However, as context-free grammars are a subset of context-sensitive grammars, the possibility of expanding existing solutions with new rules captures the context-sensitive part of the music composition. The study “Music as a Formal Language”[10] suggests that the generative capacity for representing music must be beyond that of context-free languages. The study also implies that music has features of *mildly context-sensitive language.*

When writing this thesis, a work on a probabilistic(k,l) context-sensitive grammar study was published. [11] The study uses Hidden Markov Model and Gibbs sampling method on chord sequences. Results show that Gibbs sampling can improve the perplexity of grammars.

L-Systems Lindenmayer System is a special variation of grammar, inspired by the structure of plants, where rewriting sequences can be done in parallel, etc. simple grammar with rules: $S \rightarrow AbAA \rightarrow aA \rightarrow c$

Could result in derivations: $S \rightarrow AbA \rightarrow aba$, or $S \rightarrow AbA \rightarrow cbc$

These grammars are useful as they represent self-similarity within the system, which in songs happens in chord progressions, motifs, and different parts of song repetitions.[12]

Stochastic Context-free grammars The word stochastic describes that each rule is accompanied by an activation probability. They are often used as an implementation of results from Markov models. The PCFGs are used for the synthesis of new melodies, rhythm patterns, or chord progressions. One of the studies ”Musical Composition with Stochastic Context-Free Grammars” [13] uses the structures to create a new composition. The rules are set manually by the author. In this thesis, the structure of the generator is highly inspired by this study.

1.3 Machine-learning techniques

With every year, the storage of data is cheaper, which creates large datasets in various fields. Music is no exception as *large datasets* such as the GiantMIDI-Piano [14] or Groove MIDI dataset [15] are created and available to public. The datasets can now easily contain hundreds of GB of data and still be accessible by a large number of researchers. Machine learning is an approach to *learn structures* reappearing in song and creating a model that represents a given dataset.

1.3.1 Markov chains

The discrete Markov chain is in a simple form, a stochastic process, where each probability of event *depends on the previous states* (events). The transition matrix is a representation of the model and it describes the probabilities of events based on the values in this matrix. The Markov chains can be generated only by training the model with corpus or the probabilities can be introduced by an experienced musician. The latter is often used in programs that are accompanying musicians in composing songs.

The main characteristic of the Markov chain is *a memory* (order of the model), denoting the number of states the system requires to be able to predict the next state. In the algorithmic composition, Markov chains are most often used to generate melodies or genres that have a well-described set of rules. The number of publications on melody generation using Markov chains is quite large, among those a study "Finite-Length Markov Processes with Constraints" [16], which describes the generation of melodies.

Markov chains are also used for the analysis of music structures, one of these is Hook theory [17] that used over 5000 popular songs to identify structures like chord progressions or key signatures using Markov chains. The website hooktheory.com offers a free API that has data from their analysis of approximately 5000 songs. This source is used in this work in chord progressions as predefined probabilities.

The downsides of the Markov chains are:

- Inability to capture global similarities with raw data (etc. repetitions, choruses) as higher-order Markov chains are impractical
- The generated pieces can resemble trained data.

1.3.2 Artificial neural networks

Artificial neural networks are models inspired by the biological structure found in the nervous system of animals – neurons. The individual approximations of neurons are constructed into interconnected networks. In recent years, deep neural networks are used, describing a large set of neurons assembled into layers. Typical use is in *supervised learning*, where datasets are labeled by some desired values, and the model learns to predict based on the labels.

In the process of algorithmic composition, the ANN is used with large datasets of musical data and the quality of the data is essential. The NNs can be used to compress songs, classify genres, evaluate the quality (fitness function), or produce new songs. The representation of music that is most often used as input is a piano roll, however, even successful raw sound generators were created, one of the recent developments by Deepmind – Wavenet [18]. The downside of the NNs is that the models consist of thousands of neurons and in such a vast space it is unfeasible to extract any information of *how the system operates.*

The biggest projects that use artificial neural networks are AIVA [19], MuseGAN [20] or Jukebox [21]. JukeBox is particularly interesting as it also synthesizes lyrics and applies them to the style of various singers. There are many other projects and with publicly available datasets of songs anyone can create their implementation with the use of frameworks specified for music synthesis, the well-known is Magenta [22], which is an extension of TensorFlow library.

1.4 Other approaches

Evolutionary algorithms Another approach inspired by biological processes is an evolutionary algorithm, where populations are repeatedly subjected to evaluation, selection, and reproduction, and cyclically produce a new population. Various events modifying the world can occur throughout the generations. This approach has been applied in fields where a large number of solutions can occur. The most important element of EAs is the fitness function and proper representation of the data (individual).

The representation of the data is not an issue in music, but the *fitness function* describing the quality of individuals is in terms of music quality *hard to define* and no such function has been so far found. That is why they are often combined with other models, such as ANN or used in interactive form, where a human evaluates the solutions instead of the fitness function, however, as the population is at least in tens or hundreds of individuals, this approach is quite time-consuming.

Cellular automata Is a system described by simple computational units, that operates over n-dimensional space (grid). The cells are in each time step evaluated by the transition rules. Cellular Automata (CA) tend to produce *either stationary or chaotic patterns.* In music, the CA are used for the algorithmic composition, however, the behavior of the CA is not ideal for music generation as the generated songs are either too simple or way too complex. One of the known composers using CA was Xenakis, who edited the created pieces manually to achieve more pleasing sounds.

1.5 Summary

Despite the success of the generators (such as AIVA [19][2]), people seem to be negatively biased toward computer-generated music [23] depending on their level of expertise. The general public seems to be skeptical, however, professionals admire the structures created by these systems. There are a lot of frameworks that work with large datasets and use them for machine learning. Even though it has some promising results, it is hard to learn from these models and be able to understand the structures created by these systems.

[2]AIVA even provides many songs for free to listen to or even generate your own based on few attributes.

Chapter 2

Representation of music

In this work, music is represented several ways, from the input MIDI format to the internal representation in the implementation part, to other types of musical notation used to describe the music in text and figures. These representations often describe some qualities of musical structures and usually they implement at least the most basic ones such as limited pitch range. This chapter only outlines different representations, additionally a few of their examples and a description of their use in the work.

2.1 MIDI

Musical Instrumental Digital Interface (MIDI) is a standardized binary protocol for real-time communication between musical instruments and electronic devices (computers, recorders...). It is still used in synthesizers and electronic representations of music notation despite limitations in its design. The standard defines how devices communicate and the format of the file. Instead of using a continuous stream of information (eg. sound stream), it works with a *sequence of events*. Events are messages paired with ticks that act as a timestamp. Messages are read chronologically based on the ticks. The most important messages are NoteOn and NoteOff. They signalize when the note starts to play and when it stops playing in terms of the inner ticks. These messages have several parameters:

- Integer value (0-16) – channel. It describes on which channel the given note is played. This is useful when we separate channels by either their octave span or if different channels are played by different instruments. [3] In this work, only 2 channels will be used and to separate the melody (channel(0)) and bass (channel(1))

[3]This makes MIDI files impractical for generating more complex compositions, such as music for orchestra and not for the only reason

- Integer value (0-127) – pitch of the note. The difference between pitch heights of 1 is equivalent to the distance of one semitone in a 12-tone chromatic scale. Figure 2.1lists part of the MIDI values.
- Integer value (0-100) – the volume

The standard tone A4(440Hz) has a pitch value of 69, the pitch value(p) of any tone in MIDI can be calculated by:

$$p = 69 + 12 * \log_2 \frac{f}{440\ Hz}$$

The duration of a tone is obtained by subtracting the tick value of its NoteOn message and NoteOff. To be able to do that, NoteOn events have to be matched with the corresponding NoteOff events. The duration obtained is in the number of ticks. However, "What is the length of the note in seconds?". The next message that works with the tempo of the song is the Tempo. Tempo sets the value of time (in microseconds) per quarter note – MPQ. To get Beats Per Minute (BPM)[4], which is standardly used by musicians, BPM = $\frac{60*10^6}{MPQ}$. In figure 2.3, the melody is shown in terms of musical notation and on the left, corresponding extracted MIDI events of notes.

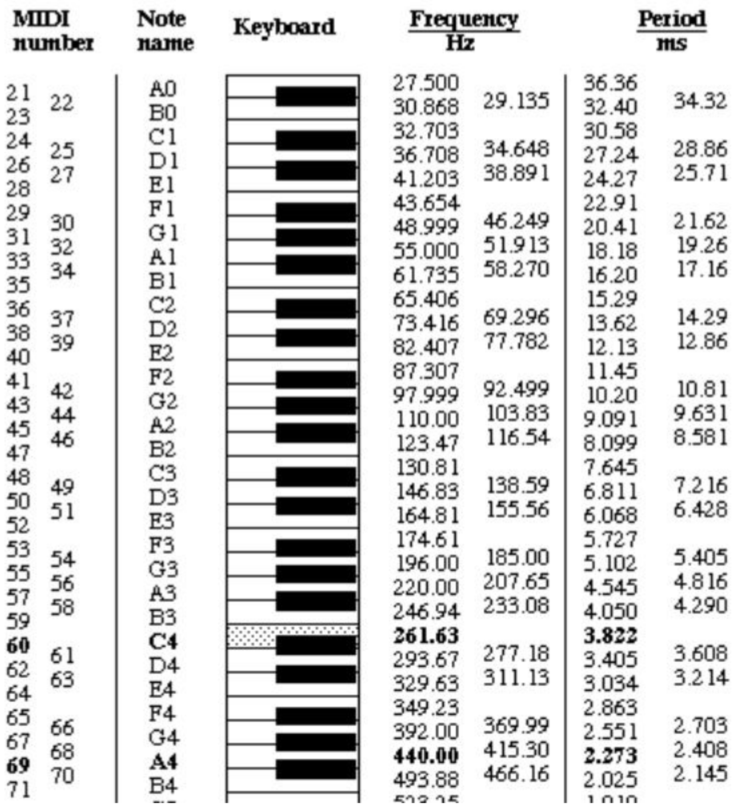
The number of ticks of the quarter note in the figure is 256, the MIDI specifies this value in the header. The value is described as Pulses Per Quarter note (PPQ) and it specifies what length of a quarter note is used for the song. The MIDI standard does not specify what PPQ value should be used, but most often the value for PPQ is 480. Unfortunately, not all songs have this value at 480 and songs have to be changed to the same value of PPQ.

MIDI has also an option to predefine the key signature of the song. However, it is optional and does not have to exist or even be incorrect. Therefore, this value will not be used, and instead, an own analysis of the song to identify the key signature of the song is used. As MIDI mostly works with NoteOn and NoteOff messages, it does not capture any dependency structures from the perspective of music theory. To acquire any information about the song, algorithms for analysis have to be developed. In section Preprocessing, the needed structures are listed and the issues arising from MIDI representation and their solutions are described.

2.2 Piano roll

The piano roll is derived from automated pianos, which originally consisted of a piano and a roll of paper with punched holes that represent the note control of the piano key. The notes had various durations based on the length of the corresponding hole. This representation is often used to visualize the content

[4]Number of quarter notes per minute



MIDI number		Note name	Frequency Hz		Period ms	
21	22	A0	27.500	29.135	36.36	34.32
23		B0	30.868		32.40	
24	25	C1	32.703	34.648	30.58	28.86
26	27	D1	36.708	38.891	27.24	25.71
28		E1	41.203		24.27	
29	30	F1	43.654	46.249	22.91	21.62
31	32	G1	48.999	51.913	20.41	19.26
33	34	A1	55.000	58.270	18.18	17.16
35		B1	61.735		16.20	
36	37	C2	65.406	69.296	15.29	14.29
38	39	D2	73.416	77.782	13.62	12.86
40		E2	82.407		12.13	
41	42	F2	87.307	92.499	11.45	10.81
43	44	G2	97.999	103.83	10.20	9.631
45	46	A2	110.00	116.54	9.091	8.581
47		B2	123.47		8.099	
48	49	C3	130.81	138.59	7.645	7.216
50	51	D3	146.83	155.56	6.811	6.428
52		E3	164.81		6.068	
53	54	F3	174.61	185.00	5.727	5.405
55	56	G3	196.00	207.65	5.102	4.816
57	58	A3	220.00	233.08	4.545	4.290
59		B3	246.94		4.050	
60	61	**C4**	**261.63**	277.18	**3.822**	3.608
62	63	D4	293.67	311.13	3.405	3.214
64		E4	329.63		3.034	
65	66	F4	349.23	369.99	2.863	2.703
67	68	G4	392.00	415.30	2.551	2.408
69	70	**A4**	**440.00**	466.16	**2.273**	2.145
71		B4	493.88		2.025	

Figure 2.1: MIDI values with analogous representations in piano, names of tones and frequencies [24]

```
[0,NoteOn(72,1,100)]
[256,NoteOff(72,1,0)]
[256,NoteOn(69,1,100)]
[768,NoteOff(69,1,0)]
[768,NoteOn(65,1,100)]
[896,NoteOff(65,1,0)]
[896,NoteOn(65,1,100)]
[1024,NoteOff(65,1,0)]
[896,NoteOn(60,1,100)]
[1024,NoteOff(60,1,0)]
```

Figure 2.2: MIDI events of simple melody

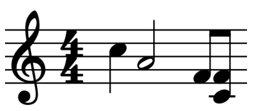


Figure 2.3: Simple melody in scientific pitch notation 2.4 to show different representation of music

of the MIDI file[5] as the events NoteOn and NoteOff can be easily transformed

[5]As the binary format is for most people unreadable

to ”holes” in the piano roll. Piano roll (shown in the figure 2.4) is formed by two axes, the vertical for pitch and the horizontal for time.

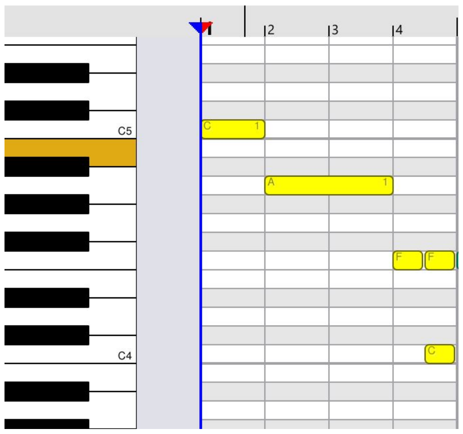


Figure 2.4: Piano roll of melody in figure 2.3, on the right corresponding piano keys are displayed

2.3 Scientific pitch notation

Is a string-based notation that specifies the pitch of notes in terms of their pitch class in 12-tone chromatic scale[6] and a number identifying the octave. The pitches are separated into octaves based on the standard tuning and the lowest note is C0 with the frequency of 16.352Hz. As an example, *A3* and *A4* are notes of the same pitch class but an octave apart. The notation is easy to use and legible in text, especially if talking about notes of unspecified duration.

2.4 Musical notation

This section describes modern staff notation(standard music notation). It captures many concepts from chapter Music theory, and therefore here we only describe the visual aspect of the notation, and definitions are omitted. The notation is a system to visually represent music, and it tries to legibly display the musical structure to musicians[7]. Besides visualizing the tones of the songs, it also captures audible characteristics of instruments and therefore it can vary depending on the instruments used in the song. [8] The notation uses a list of symbols to define structures.

[6]The chromatic scale is described in paragraph 3.1

[7]To be able to read the notation, knowledge of western music is expected

[8]such as dynamics and articulation of the song

Staff Most of the symbols are formatted on a staff. Staff is made of 5 parallel horizontal lines. The pitch of the notes is referenced to the staff and if the pitch reaches out of scope, ledger lines are used. Clefs are symbols found at the beginning of the line (or if the clef is changed). The two basic clefs are G clef(treble clef) and F clef (bass clef), with symbols 𝄞, 𝄢, respectively. G clef is spanning notes that are mainly used in melodies (approximately from C4 to C6) and the C clef describes the range of pitches used for bass (C2-C4).

Measure The staff is segmented into measures, which are of equal length of time. They correspond to the specific number of beats predefined at the begging of the song – Time signature. The division into bars provides reference points and usually, repetitions in songs are signalized at this level.

Key signature The key signature describes the range of pitches used for the song. It does not limit the number of pitches that can be used, but it refers to the scale that is used for the song. It is expected that the notes used in the song will be in the range predefined by the key signature. It consists of symbols ♭, and ♯, which describe all notes on the same horizontal line.

Chapter 3

Music theory

Music theory(MT) is a study that captures the practices of music and represents them in various structures. The structures identified can be melody, accompaniment, rhythm, harmony, tonal systems, tuning, etc. This chapter introduces the fundamental structures of music theory and describes where these structures can be found in songs.

3.1 Tone

A tone is the most basic unit in music created from sound. Sound is a vibration, therefore, it is measured in frequency and amplitude. The range of frequencies that are perceptible by people is from 20 Hz up to 15 000 Hz, but music usually spans frequencies from 50 Hz to 4000 Hz. The simplest of units in music is tone and it is defined as a periodic steady sound.

In musical terms, the frequency of the tone is said to be a pitch and the amplitude is a volume. The length of time that the sound can be heard is labeled as duration. Timbre of the tone is not important for this work as notes are projected on the piano keys and other instruments are not used.[9]. Volume in this work is also overlooked, thus in this work, the musical tone can be defined by only 2 characteristics instead of 4: pitch and duration.

Duration of sound can be of any length. In terms of music, a discrete space of note durations is used. In musical notation, the durations are represented as ratios of the given time signature. The most frequent durations and their shortcuts used are whole note(WN), half note(HN), quarter note(QN), an eighth note(EN), and sixteenth note(SN). The time signature of a song can vary, often different genres of music use different time signatures. In this work,

[9]Timbre distinguishes between different sound productions, etc. human voice, piano. This work does not work with different types of timbers and all results are in piano(the default format for MIDI)

only two time signatures are encountered, those are $frac24$ and $frac44$. The upper number describes how many beats per measure and the bottom number describes what kind of duration is a beat, 4 describes QN. [10] Therefore, with a time signature $frac44$ the number of QN in a sequence is 4. Durations can be combined, lengthened, or adjusted by special symbols in the notation. To not go too deep into the details, most of the descriptions consist of the basic 5 durations. In musical notation, the durations are displayed as:

- WN - 𝅝
- HN - 𝅗𝅥
- QN - 𝅘𝅥
- EN - 𝅘𝅥𝅮
- SN - 𝅘𝅥𝅯

Pitch A sound that is represented by a single sine wave is identified as a single pitch. Any frequency of sound can be described as a mixture of sinusoidal waves. A combination of sine waves is perceived as a single tone, and it is called a combined tone. This representation of music is way too complicated and is rarely used to generate new music. As the music developed, several relationships between tones emerged and a series of tones were created. Within these series, *pitch represents the relationship within the series.* Most often these series use natural numbers or a set of letters to describe pitches.

Harmonic series The simplest of the series is the harmonic series that has the definition described by:

$a_{n+1} = a_n + d,\ a_1 = f_0\ \wedge\ d = f_0$, where f_0 is the frequency of base tone and a_n describes frequency of nth tone in the series.

From this definition, tones from a harmonic series always progress in equidistant steps. [11] In practice to achieve better qualities of intervals (octaves, fifths, and so on) within harmonic series, inaccuracies (small deviations) from the equation are introduced.[12] The deviations in 12-tone chromatic scale are described in figure 3.1.

[10] 2 – half note beats and 8 – eighth note beats, but these will not be used

[11] Such as scales, octaves, fifths, etc.

[12] For average person unnoticeable deviations

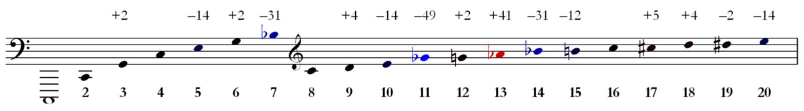


Figure 3.1: Harmonic series in notations, the numbers above the notes describe the deviations in cents [25]

12-tone chromatic scale and tuning Almost all instruments use a 12-tone chromatic scale, meaning that instruments are designed to create sounds from that set of tones. The chromatic scale is equal-tempered if the distance between subsequent notes is 100 cents. [13] The scale is therefore defined by tones that are in the ratio 2:1. Western music uses the tuning of 440 Hz, and this tone is named A4. Instead of indexing tones from the chromatic scale, a name for each tone in the series is given, these are:

$$C, C\sharp/D\flat, D, D\sharp/E\flat, E, F, F\sharp/G\flat, G, G\sharp/A\flat, A, A\sharp/B\flat, B.$$

The step between the subsequent tones is called a semitone step, and the step of two semitones is called a tone step. For clarification, D is a tone step above the tone C and C is a 2 semitone step below D in this scale. In the figure 3.2, the distance between neighboring pitches is one semitone.

The 12-tones in conjunction with a number corresponds to a specific tone in the scale, where the number depicts the octave. Therefore, A4 describes a tone that has pitch A, and it is in the 4th octave. A musical note is different from a musical tone in that tone refers to the musical notation and a tone can be characterized by more labels, but here a note and a tone will be used interchangeably.

Interval Interval describes the distance between two notes. It can refer to either the ratio of the two notes or in terms of cents. The simplest is to use numbers to describe the number of semitone steps (the distance between two successive tones) in the chromatic scale. Special names are often used to describe these intervals as they represent the quality of the interval.

interval quality The quality of the interval separates intervals into different groups based on the percept of the intervals. Identified groups are perfect, major, minor, augmented, and diminished intervals. In the next section, we describe only the most important intervals that appear in the chromatic scale.

[13]Cent is a logarithmic measure where 1200 cents is the whole octave and 100 cents is one semitone

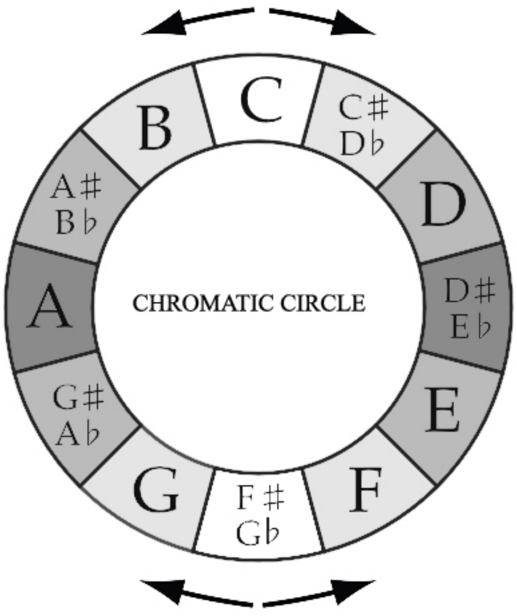


Figure 3.2: Pitch classes in chromatic circle [26]

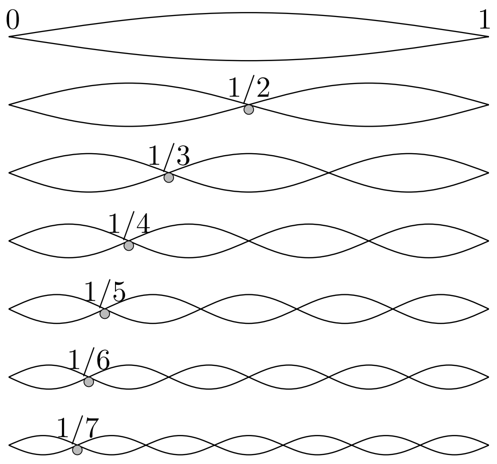


Figure 3.3: Harmonic frequencies are used in defining octaves. Two tones with ratio of frequencies 2:1 are octave apart [27]

3.2 Harmony

Harmony describes the perceptual property of music. We identify two qualities, consonance and dissonance. Consonance refers to the pleasant, beautiful sound and in contrast, dissonance brings an unpleasant and rough sense to the music. Harmony lists different types of intervals and their properties, the properties are then applied to the construction of chords. Not only simultaneously played notes experience harmony, but also the successive undergoes this property.

Consonant intervals We will discuss the simplest of consonant examples. These are perfect intervals: unison, perfect octave, perfect fourth, and the perfect fifth. Simple two-note intervals are described, and later they are followed by creating chords. The perfect intervals are typically in a small-integer ratio of their frequencies.

First are the unisons, which simply describe simultaneously played notes of the same pitch. This consonance is the simplest, in ratio 1:1. The unisons are used if multiple instruments play together, however, in this thesis only a single instrument is considered.

Octave intervals Next, the perfect octave is when two notes of the same pitch class are from adjacent octaves, for example – C3, C4. We have already mentioned that these notes share similarities from a sinusoidal perspective of sound, and that is that they are in a 2:1 ratio, this is what the figure 3.3 displays. [14]. In practice, this phenomenon is used to double voice leading and make the melody sound richer.

Perfect fifths The perfect fifth is an interval, where the frequency ratio is 3:2. Perfect fifths are abundant in musical pieces, and they are one of the key features of western music. An important structure is used in music theory called a circle of fifths, the structure is displayed in figure 3.4. Neighboring pitches are in a 3:2 ratio.

Perfect fourths These intervals have a ratio of 4:3. The perfect fourth is a complement to the perfect fifth, which means that if put in succession C $\rightarrow$ G $\rightarrow$ C (fifth followed by the fourth interval), we get the same pitch. It is important to see that we have created a chord with 3 of the perfect consonant intervals in a single chord: perfect octave, perfect fifth, and perfect fourth. This structure is often used in accompaniment.

Dissonance Dissonance appears when the ratio of the interval. Typical examples are notes one semitone apart. In this thesis, avoiding dissonance is simply achieved via expressing the songs in terms of heptatonic scales.

[14]In western music, the tuning of the keys is slightly adjusted to fit the 12-tone chromatic scale, and some of the ratios are not exactly true in the sense of the implementation in instruments, but these minor adjustments are to an untrained ear indistinguishable

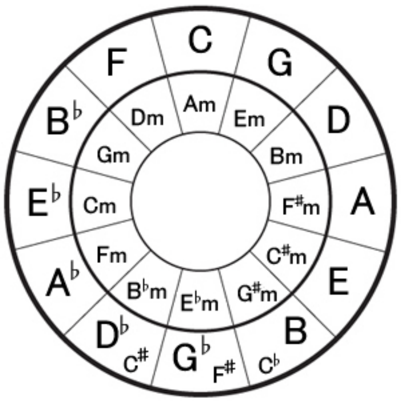


Figure 3.4: Outer pitch classes represent circle of fifths and inner the circle of fourths fourths

3.2.1 Pitch space

There were many attempts to identify different types of mathematical representations of relations between pitch classes. Different pitch spaces can abstract various features and relations between pitches. The most known, we already mentioned is a space, where the repetition of the 12-tone chromatic scale is represented, figure 3.2.

3.2.2 Chords

A chord is a harmonic set of notes or pitches. These musical structures are in their simple basic form played simultaneously, but they can even describe a sequence of notes played in succession.[15] In this thesis, the maximum number of notes in a single chord is 3. The chords are created from intervals, and they are often described in terms of scale.

The naming of the chords is skipped in this thesis as it creates unnecessary complexity to learn all naming. Instead, chords will be described as ordered tuples of interval values. For example, C major chord will be described as (C, E, G). The succession of chords will be described with a right arrow ($\rightarrow$). Instead of using the names of the chords, we will be using Roman numerals to describe the intervals of the root note. This is often used when describing the chord progression in songs.

During the research for this thesis, many representations of chord spaces were reviewed, however, no relation between a practice and their structure was found. In the book ”The Geometry of Music” [28], the main part describes chordal spaces and properties of the chords. The main property is a symmetry of chords called OPTIC symmetry. The symmetries are listed in figure 3.5. These are useful as they retain some of the qualities of the chord.

[15]Arpeggios are one of these structures, and they are used mainly in chord progressions

The OPTIC symmetries

- **O**ctave – any note from the chord can be moved by an octave
- **P**ermutation – order of the notes can be changed
- **T**ransposition – move all notes by same amount
- **I**nversion – all notes are reflected around specified note
- **C**ardinality – notes with same pitches as in chord can be added

Figure 3.5: These operations with chords retain some properties of the chords.

In this thesis, the symmetry of octave, cardinality, and transposition is used. Moreover, another symmetry is proposed by me and that is a simplification, where chords can be altered by removing their least significant note. In this thesis, the only chord type that is identified is the triad, specifically major and minor triads.

3.2.3 Triads

Triads are chords that *consist of three pitches*. If the reader is not familiar with the scales, I recommend skipping to section Scales. Triads refer to major and minor scales, and their notes are created from the intervals first (root), third, and fifth of the scale.

Therefore, in C-major(C,D,E,F,G,A,B), one of the triads could be (C,E,G) or (G,B,D). In total, 7 such triads can be created for any diatonic scale. The Roman numerals refer to the root of the chord, so classifying the chord as "I" in the C-major scale would create (C, E, G) triad and "V" (G, B, D). The numerals "I, II, III, IV, VI, VII" describe major triads, while "i, ii, iii, iv, vi, vii" minor triads. These notations simplify how individual chord progressions can be represented. In this work, I use mainly (I-VII) to describe chords in both types of scales.

As triads consist of the root, third, and fifth intervals of the scale for some chords the intervals are more significant. In the "I" chord, the root and fifth create a perfect fifth consonance, this is why for the triad I created a significance measure, the significance is a function that assigns each note of the diatonic scale a value describing its importance in the triad chord. The measure is (10, 1, 5, 1, 7, 1, 2), where the first element is the root of the chord. As an example C-major and root V would produce that pitches would be assigned values: $G = 10, A = 1, B = 5, C = 1, D = 7, E = 1, F = 2$. This is used in chord type extraction in Analyzer.

Rule		description
Cardinality	(E3,C2,E4,G3,A5)	E3,E4 remove duplicate pitch class notes
Octave	(E3,C2,G3,A5)	closest C to E3 is C3, same with G and A
Permutation	(E3,C3,G3,A3)	C3 is lowest, then E then G then A

Table 3.1: Example of chord simplification on (E3,C2,E4,G3,A5)

3.2.4 Chord Progression

Songs typically consist of two parallel sets of notes, melody, and accompaniment. An accompaniment is the underlying structure of the song that makes the song sound more ”richer”.

The ”beat” is a repeating rhythmical pattern of the accompaniment, and the chord progression is a sequence of chord types. Despite the differences in the words ”bass”, ”chord progression”, and ”accompaniment” I use them interchangeably to describe the underlying structure of the song.

The representation of chord progressions is as a sequence of Roman numerals. If the song is written on a diatonic scale, I-V would describe the progression of the first and fifth chord. The chord progressions are usually of size 3 or 4, depending on the genre and complexity of the composition.

In the practical part, we will focus a lot on chord progressions as they provide a lot of information about the bass. Therefore, for the analysis, we will have to identify each chord’s root key. We already discussed the OPTIC symmetries, but for identification, we only need 3 of these transformations. Octave, permutation, and cardinality. To limit the number of chords in the song, the chords are transformed to a minimal form by the OPC rules(from the OPTIC symmetries). As an example, description of how the chord (E3, C2, E4, G3, A5) is transformed to (C3, E3, G3, A3) in table 3.1.

In chord progressions, chords typically change in fixed durations, which are of length WN or HN. The number of different chord progressions in songs is really small as the chord progressions are repeating. The figure 3.6 describes allowed progressions of chords for major and minor scales described in GOM.[16] In some genres, such as blues, the chord progressions are even more strictly limited.

The diagram defined in the figure can be translated to a state transition system if considering the Roman numerals as states and arrows are the transitions to the next state. In this way, allowed major chord progressions are described by the STS in the table 3.2. This state transition system could be used to generate new chord progressions by traversing the states. For exam-

[16]The exceptions of songs using different types of progression can occur, however in practice majority of the songs undergo these rules, the same applies to the following rules.

ple, one possible chord progression generated is I-ii-V-IV-I in the chords of the C-major scale, C-dm-G-F-C.

In the analysis of 1300 popular songs by Dave Carlton [17] a lot of information about the chord progressions was found, for example, if the song started with a C major chord, the following chord was with 31% probability G major. This is not a surprise as the building blocks of chord progressions in popular music are I, V, and IV chords.

In the section Analyzer, the probabilities from the analysis of popular songs are used to create a model of chord progression. The chord progressions of popular songs were created via 1st-order Markov chains, and the probabilities can be found at *hooktheory.com* with visual representation. The collected probabilities are summed in the table 3.3, only probabilities of more than 5% are collected, and in the practical part, the table is normalized.

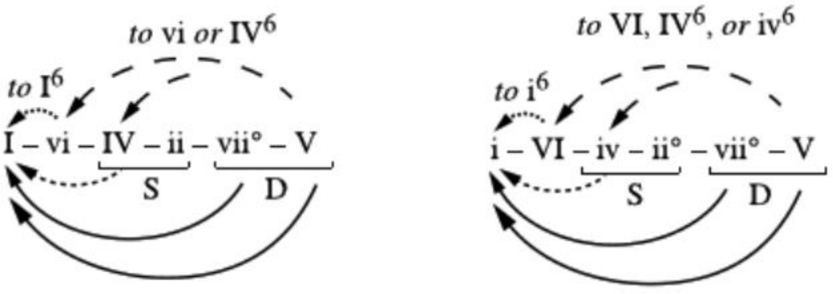


Figure 3.6: Allowed chord progressions in major (left) and minor(right) scales, figure from the book "A geometry of music" [28]

I →vi | S | D
vi →S | D | I
S →IV | ii
IV→ii | D |I
D →vii | V
vii→V | I
V →vi | IV | I

Table 3.2: Rules of grammar, describing the allowed major chord progressions

3.2.5 Scales

Scales represent a set of pitch profiles. Usually, the scales are heptatonic[17], meaning that scale is constructed from 7 pitch classes of 12-tone chromatic scale. They are most often described as a set of pitch classes, for example, C

[17]octave repeating

chord	I	ii	iii	IV	V	vi
I	-	0.06	0.04	0.182	0.252	0.1
ii	0.06	-	0.081	0.155	0.064	0.063
iii	0.04	0.081	-	0.326	0.077	0.06
IV	0.182	0.155	0.326	-	0.205	0.232
V	0.252	0.064	0.077	0.205	-	0.26
vi	0.1	0.063	0.06	0.232	0.26	-

Table 3.3: Probabilities of the most significant chords in popular songs, collected from HookTheory, the code to extract probabilities is in attachments

major scale {C, D, E, F, G, A, B, C}, but they represent the whole range of notes as the pattern repeats for each octave.

In musical notation, scales are predefined at the beginning of the song by a key signature. Songs are written in terms of scales, and most of the notes present in the composition belong to the predefined scale. Using notes out of this scope creates accidentals, which produces dissonance. In songs many scales can be found, however, they do not overlap, meaning only one scale is used simultaneously. In this thesis, only single-scale compositions are examined.

It is possible to represent the scales in several ways. The most common is using a sequence of pitches, where the pitches are in ascending order (for example, C major scale (C, D, E, F, G, A, B)). Other representations could be obtained with a root pitch (lowest pitch note) and a list of intervals from the given note corresponding to the interval in a 12-tone chromatic scale.

Scale types Music theory describes many scale types that are varying in span and number of notes. They are usually tightly connected to the culture or the specific genre of music. In this thesis, only major and minor scale types are used as they are the most abundant and easier to classify. [18]

Major Most used scale in Western music, which evokes consonance. Steps of this scale are (2,2,1,2,2,2,1)[19], which can be transformed to scale type (0,2,4,5,7,9,11,12). Therefore, the major scales are (r, (0,2,4,5,7,9,11,12), where r is the root pitch of the scale

Minor Songs in minor scales sound melancholic compared to major. Steps of minor scale are (2,1,2,2,1,2,2), and transformed to scale definition - (r,(0,2,3,5,7,8,10,12)).

[18]It is possible to extend the current implementation to work with different scale types too, however besides the interval list and root note, the profile is also needed for key signature analysis of songs

[19]2 - is Whole step, 1- is Half step

Comparison of scale types For each minor scale, there exists a major scale that has the same pitch range. As an example, figure 3.7 shows two scales, C-major and A-minor with notes (C,D,E,F,G,A,B) and (A,B,C,D,E,F,G) respectively. To capture the pitch equivalence between major and minor scales, the circle of fifths/fourths (figure 3.4) is used, where the outer circle represents major and the inner circle minor scales. The scales next to each other are scales with the same pitch profile.

The difference is between the two scales is in the order of the pitches, which changes the intervals within the scale. It is best seen in 3.2.4, where the degrees are used to construct chords. A similar scheme of I-V-IV would be different major and minor scales. As an example, in C-major the intervals would be C-G-F, and in A-minor, A-E-D.

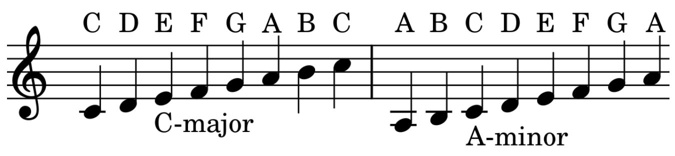


Figure 3.7: C-major and A-minor scales

Distance I define a measure of the distance built from the circle of fifths. For better reference, the circle of fifths is transformed to a lattice so that the minor scales are at one vertical level and major at the other (figure 3.8. In the lattice, the distance between horizontal neighbors is 1.

Two scales are considered equal if their pitch profiles are the same, describing the vertical edge.

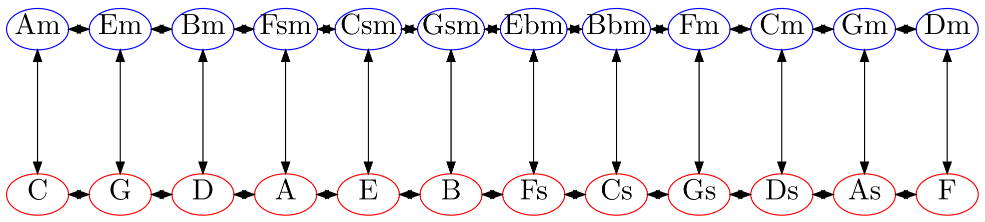


Figure 3.8: Lattice of scales, the upper level is minor and lower major.

3.3 Tonality

It is a principle of building music around the central note, often described as tonic. We identify two types of tonality major-minor, which refer to the scales used. I have already mentioned many features of tonality in section Harmony, thus, I will not go into too many details of the tonality theory. Only 5 features contributing to tonality are reviewed.

Melody A melody is a series of tones that are played in succession. The melody is one of the key aspects of music and its construction is a primary interest in music theory. Melodies often consist of several motifs that are reappearing across the whole song. If notes of the melody do not overlap (sound at the same time), the melody is monophonic.

3.3.1 Features of tonality

One of the composers and music theorists dealing with musical structure and tonality is Dmitri Tymoczko. He wrote many books on this topic and one of them was used as a knowledge base for this thesis. This section describes features from the book ”A Geometry of Music” [28]. The 5 features of tonality are:

- Conjunct melodic motion
- Acoustic consonance
- Harmonic consistency
- Limited macroharmony
- Centricity

Conjunct melodic motion It describes a melody and refers to the preference of small steps instead of larger ones even though the interval has better quality. For example, octaves and fifths are highly consonant intervals, but instead of going immediately to these intervals, the preferred movement is going to be smaller, etc. second interval and then to fifth. This is also present in constructing arpeggios, where one octave is divided[20] and the notes are played in small steps.

Acoustic consonance Is the description of interval properties. The intervals were already reviewed in section Interval

Harmonic consistency This feature describes long-term relationships between structures in the song. The structures that are similar to each other (chord type, rhythm pattern) create an audible similarity.

Limited macroharmony Songs typically consist of a small set of notes. A song can be divided based on the reappearing motifs and the repetition of the same chord progression. Some of the songs have a small number of these sections, which can be identified as chorus-verse schemes. Other abundant

[20]Preferably evenly

parts present in songs are bridges[21], intro[22] and outro[23] In the generator this is applied by generating the transition song in 3 sections – intro, middle, outro.

Centricity Some of the notes or pitches tend to be more dominant in the song, many songs have a central point of the song to which it repeatedly returns. Often songs start and end with central notes. This feature is utilized in generating a generic melody.

[21]They connect chorus and verse

[22]First few measures of a song

[23]Several last measures of the song

Chapter 4

Computational background in music

This chapter focuses on the mathematical and computational models used for the analysis and synthesis of the song. The main principle is a stochastic generation, where the model describes the music structure in terms of events and assigns probabilities to each event.

4.1 Stochastic generation

The probabilistic models to be used for automatic generation are mainly stochastic state transition systems. These are transition systems that for each transition a probability is assigned. The probabilities can be predefined during construction or learned based on the events from the data set. The generator works with a set of events that it uses to represent the data.

4.1.1 Event Space

When working with probabilities, it is necessary to define an event space, which describes what parameters and what range of values is to be used. There are many measurable features in music theory, and it is possible to use any of the musical sets in an event space. However, with each attribute, the complexity of the generator increases.

Definition 4.1.1 *Event space is an ordered set of events* $E = (e_1, e_2, /dots, e_n)$
E is the event space , where event represents values of the generator e_i *- e is an individual event that*

In this thesis, event space is represented with various structres from music theory. For example, we will list a few here:

$E_1 =<$ 𝅝, 𝅗𝅥, ♩, ♪ $>$
as a representation of duration

or we can represent it in terms of Double as
$E_2 =< 1, 0.5, 0.25, 0.125 >$
where 1 is a WN, 0.5 HN, 0.25 QN and 0.125 EN

and event space of pitches: $E_3 =< 0, 1, 2, \dots, 11 >$
where value represents the pitch class of 12-tone chromatic scale

The event space can also be defined by describing events or rules defining the whole set. Such as the event space of all piano keys or a set of all one-bar rhythms.

4.1.2 Probability

Probability describes the likelihood of an event to occur in an event space based on some experiment. The probability of an event can be anywhere between 0 and 1.

In terms of music, one of the simplest probabilities could be using event space of all pitches of notes of the song, the experiment would be looking at each note and count how many times a pitch appears in the song. From the simple melody of (C3,D3,E3,D3,C4), the probabilities for p(C) $= \frac{2}{5}$, p(D) $=\frac{2}{5}$, and p(D) $= \frac{1}{5}$. To be able from these probabilities select one event at random. We need a random generator or random selection.

4.1.3 Random number generators in computers

In computers, the RNGs can be identified as two types, truly *hardware-random number generator*, which generate randomness based on some physical environmental value, and *pseudo-random number generators*, that are described by functions. In this work, only the latter generator is considered as it is the most common type. All programming languages offer libraries that implement these random generators. In this thesis, "scala.util.random" library is used to create randomness for the generators.

The randomness is pseudo-random, which means that the generator seems to be random (without an observable pattern), however, they are described as a function internally. This is often implemented via a pair of seed and RNG, where the seed has to be hidden to call the process random. Seed describes the first value of the generator. Typically the RNG is implemented via function $X_{n+1} = (aX_n + b) \bmod m$, where a, b, m are large integers and X_i is a generated number, and the seed is the X_0.[24]

The possible solutions to implement random selection from probability distributions can be various. I do not like to work with double values as they are not precise, instead the probabilities are expressed in integer values by multiplication by 10^x, typically 3 as more precision is not needed. The method uses a cumulative distribution function, but instead of the range $< 0, 1 >$, range $< 0, 10^x >$ is used. From the cumulative distribution function, an interval of integer values is extracted for each step of the function. As an example, $p(e_1) = 0.5, p(e_2) = 0.2, p(e_3) = 0.3$, would create range of values $e_1 = (0, 500 >, e_2 = (500, 700 >, e_3 = (700, 1000 >$. To obtain one event, RNG would then be used to generate a number from the range 1 to 1000, and the number would select one of the 3 events, eg. rn=233 $\rightarrow e_1$.

One more type of random selection from the set is used. This approach is applied when a small number of events are present (less than 10). The method creates a new sequence of events from the probabilities by adding each event(e) $p(e) * 10^x$ times to the sequence. Random selection is then done with a random permutation of the sequence and selecting the first event. For the $P(E) : p(e_1) = 0.5, p(e_2) = 0.2, p(e_3) = 0.3$, the sequence with $x = 1$ would be $(e_1, e_1, e_1, e_2, e_2, e_3, e_3, e_3)$. This type of selection is ineffective, however, it is more illustrative.

4.1.4 Probability generation

The probability distribution is a function that gives each event a probability. The simplest probabilistic distribution is an equally likely distribution (discrete uniform). Considering the event space E_1 the probabilities would be $p(e) = \frac{1}{4}, \forall e \in E_1$. If we were to generate events from this distribution, for each selection, we would randomly with a given probability select that item. The different types of distributions appearing in the music will not be much of a concern, however, for the generic melody generator probabilities are predefined. The probability distribution is created with a knowledge of some of the structures found in music.

Experiment-driven probability (learned) distribution is more common, as the probabilities are created from real observable situations. To find the probability distribution $P : E- > \mathbb{R}$ we could for example have an event space E_1, and our experiment would be the next occurrence of any note of given duration in simple rhythm: $P : E_1- > \mathbb{R} : P(E_1) = 1, P(𝅝) = 0.2, P(𝅗𝅥) = 0.3, P(♩) = 0.5, P(♪) = 0$, 0 probability events can be omitted from definition, by default all undefined will be 0.[25]

[24]the values of the function in library used are m=2^48 ,a=25214903917 ,b=11

[25]The likelihood of whole event space has to be 1 as the experiment is defined on given events, if different values of durations would be in the experiment, they would not be used for the probability distribution of E_1

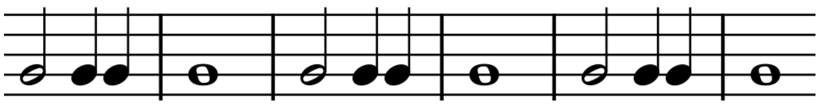


Figure 4.1: simple rhythm of song

Now, given a task to generate x number of note durations, we could generate it using these probabilities. With the 50% probability, we would get QN. Therefore, with these probabilities, one possible generated part of length 7 of the rhythm could be HN, QN, HN, QN, WN, QN, QN. Compared to simple rhythm, it easy to see that the generated sequence does not resemble any of the previous bars, thus creating a completely new rhythm. This in itself is not an issue as the generation of new sequences is often desired. It depends if the generated sequences are valid in terms of music and if this structure could appear. The sequence is valid, however, if we would want to split this sequence into bars (of 4/4 time signature), it would be impossible to do without some notes spanning two bars, which in chord progressions is not typical. To have a more restrictive generation, we have to either define a better event space that we use or implement a different type of generator.

One of the possible more complex structures could be $E_5 =< allsequencesoflengthofonebar >$, (etc.,sequence ♪,♩, ♪, 𝅗𝅥). To show the importance of selecting right event space, consider the following two experiments:

- Experiment1: $E_4 =< allsequencesof3successivenotes >$ Occurrence of an event in the song, with steps of 1 note
- Experiment2: $E_5 =< allsequencesoflengthofonebar >$ Occurrence of an event, with every bar in the song

Using these experiments on simple rhythm, probability distribution acquired for Experiment1 is:

$P_3 : E_4 \to E_4 :$
$P(e_1) = 0.3, e_1 = \{\text{𝅗𝅥}, \text{♩}, \text{♩}\},$
$P(e_2) = 0.3, e_2 = \{\text{♩}, \text{♩}, \text{𝅝}\},$
$P(e_3) = 0.2, e_3 = \{\text{♩}, \text{𝅝}, \text{𝅗𝅥}\},$
$P(e_4) = 0.2, e_4 = \{\text{𝅝}, \text{𝅗𝅥}, \text{♩}\}$

for experiment2:

$P_4 : E_5 \rightarrow E_5 :$
$P(e_1) = 0.5, e_1 =$ {𝅗𝅥, ♩, ♩},
$P(e_2) = 0.5, e_2 =$ {𝅝},
}

In the table 4.1, are values that were generated using probabilities from the experiments, the generated sequences are of various lengths to match each other in the number of notes, we can see that with the same input (example song) the results generated can be completely different for even such small variations.

Experiment	Probabilities	generated sequence
1	$e_1 = 0.2, e_2 = 0.3, e_3 = 0.5$	<𝅝, 𝅗𝅥, ♩, ♩, 𝅗𝅥>
2	$e_1 = 0.3, e_2 = 0.3, e_3 = 0.2, e_4 = 0.2$	< { ♩, ♩, 𝅝}, {𝅗𝅥, ♩, ♩}, { ♩, 𝅝, 𝅗𝅥}, { 𝅝, 𝅗𝅥, ♩}, { 𝅝, 𝅗𝅥, ♩} >
3	$e_1 = 0.5, e_2 = 0.5$	<{ 𝅝} ,{ 𝅗𝅥, ♩, ♩}, { 𝅗𝅥, ♩, ♩}, { 𝅗𝅥, ♩, ♩},{ 𝅝} >

Table 4.1: Experiments with generated sequences

Possible statistical views on symbolic musical data in the thesis ”Probabilistic Models for Music”[29] are listed. The probabilistic models are separated into sections based on the parts they represent, such as chord progressions, rhythms, melodies...
In ”Harmonic analysis with probabilistic graphical models” [30] suggest that the use of only pitch and rhythm substantially loses some information, however, the human listener can still base harmonic analysis only on these two elements. Their experiment consisted of using chords with 3 values, Tonic (the root of scale), scale (major/minor), and degree(I,..., VII).

4.1.5 Correlation

It is a statistical association of two random variables.[26]. Correlation is used to quantify various relationships between the two variables.

Definition 4.1.2 *Correlation coefficient R of two random variables X, Y is defined as*

$$corr(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

[26]when talking about correlation, it is meant Pearson correlation coefficient

where $x \in X$ and $y \in Y$ and $\overline{x}$ is the mean of all x.

The correlation has a symmetry property meaning that $corr(X, Y) = corr(Y, X)$. The correlation is a value between -1 and 1, describing linear dependence between variables. In music terms, we use the correlation for two types of analysis, scales and chords.

4.1.6 Key Detection - Scale detection

Identifying the key signature of music is important for replicating the style of music generated. The key typically limits the 12 pitches of the chromatic scale to some subset (7 in heptatonic scales). The following is a description of "The Krumhansl-Schmuckler Key-Finding Algorithm" [31]

Pitch profile The pitch profile describes the distribution of notes in the song if only the pitches of notes are considered. From the profile, it is possible to infer what type of scale is used for a song. To acquire the profile, the pitch of each note is extracted and a histogram of pitches is created. In the figure 4.2, the pitch profile of the song "Arrival of the birds" is extracted. Each bar describes the number of notes in the pitch, from this a guess of identifying that the pitches F♯/G♭, C♯/D♭, and D♯/E♭are probably significant notes of the song.[27] From this, as a musician with years of experience, I could guess that the scale is either F♯-major or G♭-major. The question "How do we create such a guess algorithmically?".

The algorithm works with correlation and compares major and minor profiles to that of a given song. The probabilities of the profiles can be various, however, I used the ones that were defined in their research. The profiles for major and minor are listed in table 4.2. For major, the significant intervals have values: root – 6.35, 5th – 5.19, and 2nd – 4.38. These intervals have the highest values and are a consequence of being abundant both in chord progressions and in the melodies.

Interval	0	1	2	3	4	5	6	7	8	9	10	11
Major	6.35	2.23	3.48	2.33	4.38	4.09	2.52	5.19	2.39	3.66	2.29	2.88
Minor	6.33	2.68	3.52	5.38	2.6	3.53	2.54	4.75	3.98	2.69	3.34	3.17
song	107	169	60	224	97	209	291	47	139	61	177	111

Table 4.2: The profiles for major and minor scale, and song Arrival of the birds [32]

To identify the scale of a song, the profile of the song has to be extracted. Then for each major and minor profile (C-major, C-minor, D-major, D-minor,

[27] contributing to chord used often or intervals that are abundant in songs, F is left out as it is too close to the most abundant pitch

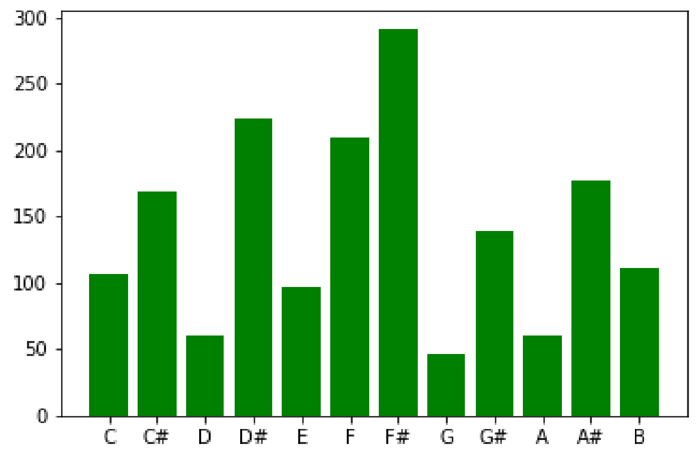


Figure 4.2: Pitch profile of the Arrival of the birds song, where y axis describes number of pitches in the song

...), the correlation coefficient is calculated. These coefficients are later compared and the maximum value is selected, which marks the most likely profile. [28]

In this thesis, a similar approach as scale identification is used to identify what type of chord is used in chord progressions. The values are closely related to the ones used in the table 4.2 with the erasure of the notes out of the scope of the identified scale. The implementation of the key finding algorithm and chord type is described in section Analyzer.

4.2 State transition systems

State transition systems are defined as a pair $(S, \rightarrow)$ where S is set of states and $\rightarrow \subset S \times S$ – transitions. STS are similar to directed graphs, where states are nodes and transitions edges. In terms of music, the states could be created similarly as events were in the previous section, and the transitions could describe the preceding event in the song. A simple melody (C4,D4,E4,E4,C4,D3), where states are pitches {C,D,E} would create transitions (C,D),(D,E),(E,E),(E,C). In figure 4.3 the created system is presented as a directed graph. This structure could be used to generate new sequences by traversing the states. One of the possible traversing, where the starting state is E, could create a melody (E, E, E, C, D).

[28]In this thesis only major and minor scales are considered, the implementation of other scales especially those spanning more than a single octave would be more challenging

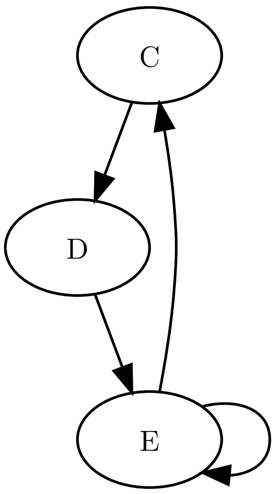


Figure 4.3: State transition system/directed graph of melody (C4,D4,E4,C4,D3)

Ergodicity of a system expresses that from every state of the system there exists a path (sequence of transitions) to every other state. For the example this property is true, and that is why the traversing of the states could have been infinite. In terms of the generators, we want to have every system ergodic as it is possible to find from any of the starting states to have a path that is of any size.

Labelled transition systems The difference from the transition systems is that each transition is labeled (from a set of labels). Therefore, they are defined by $(S, \Lambda, \rightarrow)$, where S is set of states, Λ set of labels, and $\rightarrow (S \times \Lambda \times S)$. Labels can represent different things, such as expected input, output when traversing, probability, or conditions to trigger the transition....

Deterministic property describes that for any state s and label l, there exists only one possible transition (s,l,q). It means that at any given moment (state) the label fully determines the next transition. In a non-deterministic system, for some state, there exists more than one transition with the same label. In stochastic generation, typically non-labeled, non-deterministic systems are used and at each transition, the probability describes the next action.

4.2.1 Formal grammars

Describes how to form a language (a set of words) from an alphabet. Grammars can be described as a set of production rules working on strings. The strings are generated by using a set of rules repeatedly, in so-called derivation steps. Noam Chomsky formalized generative grammars into 4 types, these are regular, context-free, context-sensitive, and formal (unrestricted) grammars, the relationship between types is regular grammars $\in$ context-free $\in$ context-sensitive $\in$ formal.

Definition 4.2.1 *Formal grammar is a quadruple (N, T, P, S)*

- *N is a finite set of nonterminal symbols*
- *T is a finite set of terminal symbols*
- *P is a set of production rules in the form $(T \cup N)^* N (T \cup N)^* \rightarrow (T \cup N)^*$, and * describes any number from set symbols preceding the symbol*
- *S a start symbol, $S \in N$*

derivation In term of grammar we identify operation derivation ($\Rightarrow$), that describes the application of grammar rules to generate string. If we always apply rules of the leftmost nonterminal first, we say that the derivation is leftmost. Often $\overset{x}{\Rightarrow}$ describes that the x rule was used during the derivation step, if multiple steps are used, a list of rules can be used above the arrow. A string is *accepted/generated* when no nonterminal is present in it and thus the derivation step cannot be applied.

In terms of music, grammars can be used to describe different parts of the song and abstract more complex structures of music. As an example grammar G_1 defined in figure 4.4

The grammar G_1 describes a possible grammar for developing chord progressions of a song. The terminal symbols describe the type of chord used and nonterminals describe the section of a song. One possible derivation Intro $\xRightarrow{1,3,4,2,3,4,3,5}$ I I V IV VI V II I V IV II I VI V II I V IV II I V II I IV V I I, creates chord progression that could be used in the song. In the figure 4.5 intermediate derivation steps are listed.

4.2.2 Context-free grammar

From the definition of formal grammars, the only difference is in the definition of P – a set of production rules, where the rules are of format $N \rightarrow (T \cup N)^*$.

In figure 4.6 the grammar from figure 4.4 is changed so that it is CFG by elimination of the conflicting 6th rule and creating a new one that describes the same rule.

4.2.3 Probabilistic context-free grammar

Or also stochastic context-free grammars are extensions of CFG, where each rule is associated with a probability. These probabilities can be either learned or given. The sum of all probabilities of nonterminal rules has to be 1. A

G_1 = (N, T, P, S)

- N = {Intro, Verse, Chorus, Bridge, Outro}
- T = {I, II, III, IV, V, VI, VII}
- S = Intro
- P = {

$$\begin{array}{ll}
1.\ Intro & \rightarrow I\ I\ V\ IV\ Verse\ Outro \\
2.\ Verse & \rightarrow Verse\ Chorus\ Verse \\
3.\ Verse & \rightarrow VI\ V\ II\ I \\
4.\ Chorus & \rightarrow V\ IV\ II\ I \\
5.\ Outro & \rightarrow IV\ V\ I\ I \\
6.\ Verse\ Chorus & \rightarrow Verse\ Bridge\ Chorus \\
7.\ Bridge & \rightarrow I\ VII
\end{array}$$

}

Figure 4.4: Formal grammar describing chord progression scheme that can be in songs

$$\begin{aligned}
Intro &\overset{1}{\Rightarrow} I\ I\ V\ IV\ Verse\ Outro \\
&\overset{3}{\Rightarrow} I\ I\ V\ IV\ VI\ V\ II\ I\ Chorus\ Verse\ Outro \\
&\overset{4}{\Rightarrow} I\ I\ V\ IV\ VI\ V\ II\ I\ V\ IV\ II\ I\ Verse\ Outro \\
&\overset{2}{\Rightarrow} I\ I\ V\ IV\ VI\ V\ II\ I\ V\ IV\ II\ I\ Verse\ Chorus\ Verse\ Outro \\
&\xRightarrow{3,4,3} I\ I\ V\ IV\ VI\ V\ II\ I\ V\ IV\ II\ I\ VI\ V\ II\ I\ V\ IV\ II\ I\ V\ II\ I\ Outro \\
&\overset{5}{\Rightarrow} I\ I\ V\ IV\ VI\ V\ II\ I\ V\ IV\ II\ I\ VI\ V\ II\ I\ V\ IV\ II\ I\ V\ II\ I\ IV\ V\ I\ I
\end{aligned}$$

Figure 4.5: Derivation steps of G_1

derivation of a string is done with the help of the random generator which selects rules based on their probabilities.

G_2 = (N, T, P, S)

- N = {Intro, Verse, Chorus, Bridge, Outro}
- T = {I, II, III, IV, V, VI, VII}
- S = Intro
- P = {

$$\begin{aligned}
&1.\ Intro \rightarrow I\ I\ V\ IV\ Verse\ Outro\\
&2.\ Verse \rightarrow Verse\ Chorus\ Verse\\
&3.\ Verse \rightarrow VI\ V\ II\ I\\
&4.\ Chorus \rightarrow V\ IV\ II\ I\\
&5.\ Outro \rightarrow IV\ V\ I\ I\\
&6.\ Verse \rightarrow Verse\ Bridge\ Chorus\ Verse\\
&7.\ Bridge \rightarrow I\ VII
\end{aligned}$$

}

Figure 4.6: Context-free grammar created from G_1 in figure 4.4

Definition 4.2.2 *Probabilistic context-free grammars are defined as a quintuple*
G=(N,T,R,S,P)
where

- *M – a set of non-terminal symbols,*
- *T – a set of terminal symbols (alphabet)*
- *R – a set of production rules*
- *S – a starting non-terminal symbol*
- *P – a function, where every production rule is assigned with a probability*

In the figure 4.7 an extension of the context-free grammar G_2 with added probabilities for each rule. This grammar is quite simple as only nonterminal *Verse* has multiple rules.

R =

$$
\begin{aligned}
p(1) &= 1\\
p(2) &= 0.5\\
p(3) &= 0.3\\
p(4) &= 1\\
p(5) &= 1\\
p(6) &= 0.2\\
p(7) &= 1
\end{aligned}
$$

Figure 4.7: Extension of CFG, by assigning each production rule a probability

4.2.4 Markov Chains

A Discrete-time Markov chain is a stochastic model, where the probabilities are learned from a sequence of events, and it describes a sequence of possible events in terms of the event obtained from a previous event. Markov chains were built to analyze the text and generate similar text based on the input. The definition of discrete-time Markov chain is defined as:

Markov chain
Given random variables $X_1, X_2, \dots, X_N, where X_i \in L$ and probability defined as $P(X_{n+1} = x | X_1 = x_1, x_2, \dots, x_n) = P(X_{n+1} = x | X_n = x_n)$

Time-homogeneous Markov chain
$P(x_{n+1} | x_1, \dots, x_n) = P(x_{n+1} | x_n)$

Markov chain with memory
$P(X_n = x_n | X_{n-1}, \dots, X_1 = x_1) = P(X_n = x_n | X_{n-1} = x_{n-1}, \dots, X_{n-m} = x_{n-m})$

In this thesis, several Markov chains are used to analyze different aspects of the song, different types are selected depending on their properties, and event spaces are built for the analysis. The Markov chain with memory is possible to be translated to a homogeneous Markov chain if we construct chains with $Y_n =< X_n, X_{n-1}, \dots, X_{n-m+1} >$, and this property is applied when generating melody notes. By convention, we assume that the Markov chain has always a transition from any state.

Transition Matrix Is a 2-dimensional matrix describing the probabilities of transitions. Let $E = (e_1, \ldots, e_n)$ be an event space that the Markov chain works with. Given transition matrix P, build on E, the probabilities of the the transition $e_i \to e_j$ is a value at index $P_{i,j}$. An element-wise sum across all rows has to be equal to 1.

$$\mathrm{P} = \begin{bmatrix} P_{1,1} & \dots & P_{1,j} & \dots & P_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{i,1} & \dots & P_{i,j} & \dots & P_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{n,1} & \dots & P_{n,j} & \dots & P_{S,n} \end{bmatrix},$$

$$\forall i, 1 \leq i \leq n, \sum_{j=1}^{n} P_{i,j} = 1$$

The probabilities are characterized by a state space, describing the probabilities of transitions. Two states are said to be communicating with each other if there is a sequence of transitions that leads to the second. For us, it will be important that for every state there exists at least one transition. In this thesis, it is achieved by appending m events from the start of the song to the end, making the whole system ergodic, this is expected for every training in the system.

Training The transition matrix for Markov chains is created based on the learning from the sample data. The sample data is transformed into a sequence of events, and then all probabilities of the transition matrix are updated. The implementation is done via iteration over a sequence of events and counting the abundance of transitions in the sequence. Therefore, if in the sequence at position 3 is e_4 and at position 4 e_3 the transition matrix is updated by 1, this is done for the whole sequence. The pseudocode in figure 4.8 describes how to obtain the probabilities.

Starting state When using a transition matrix, it is necessary to provide a starting state to generate the next sequence. As the matrix created is always ergodic, it is always possible to find the transition. In the system, a fixed starting symbol is used in the chord progression, and others use random states from the transition matrix.

Examples of Markov chains in music In music, the Markov chains are applied to many structures. As an example for Markov chains of various orders, I will be using an event space E ={1,2,4,8}, corresponding to the durations. Figure 4.9 is the sample sequence of 8 bar length. The rhythm

```
input: sequence of events
train:
    segments <- split data into order sized chunks
    for every segment:
        key <- take all events from segment except the last one
        next <- last event of a segment
        update transition matrix:
            key describes the row of the table
            next describes the column
            update cell of the transition matrix by adding weighted
                    value to the cell value

normalize:
    for every row of transition matrix:
        sum <- sum all values of row
        update every cell of the row by dividing
                    the cell value by the sum
```

Figure 4.8: Pseudo-code of obtaining transition matrix from sequence of events

contains reappearing patterns, specifically 1st, 3rd, and 6th measure have the same rhythm, similarly 4th, 7th bars and 5th with 8th. If more than 1 bar similarity is considered then bars {3, 4, 5} are the same as {6, 7, 8}. Instead of listing transition matrices for all Markov chains, I use directed graphs for easier visualization.

The simplest of Markov chains is 1st order. The transition matrix is trained using the sample rhythm and after the matrix is normalized. The resulting Markov chain is displayed in figure 4.10. This chain is ergodic, and if used as a generator of event 1, the generated sequence of length 7 is (1,1,4,2,4,8,2). Higher-order Markov chains get complicated quite quickly, therefore for 2nd order Markov. Already for this order, the transition matrix has 10 states and 14 different non-zero transitions. The generated model can be seen in figure 4.11.

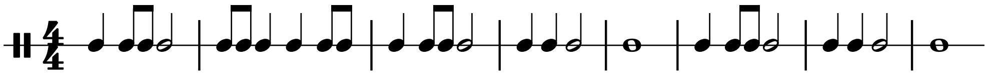


Figure 4.9: Sample rhythm of 8 measures in $\frac{4}{4}$ rhythm. In event space E, it is (2,1,1,4,1,1,2,2,1,1,2,1,1,4,2,2,4,8,2,1,1,4,2,2,4,8).

When working with durations, it is reasonable to split the data into measures. Not only the rhythm patterns generated by MM will be more similar to the input data, but also the size of possible states will be much smaller.

In the sample rhythm, we have 8 measures, however, only 4 unique rhythm

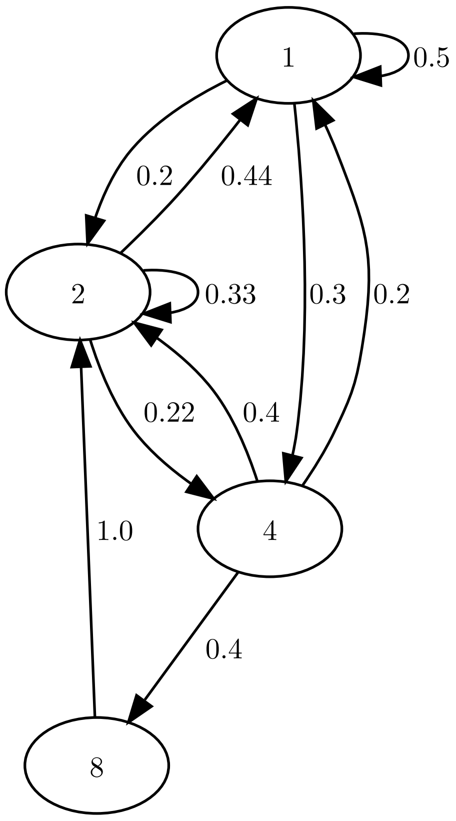


Figure 4.10: Graph of 1st order Markov chain with event space E ={1,2,4,8}

patterns in these measures. The event space E = $\{e_1 = (2,1,1,4), e_2 = (1,1,2,2,1,1), e_3 = (2,2,4), e_4(8)\}$ describes unique bars of the sample rhythm. Expressing the sample rhythm in terms of this event space results to a sequence $(e_1, e_2, e_1, e_3, e_4, e_1, e_3, e_4)$. With these, 1^{st} order Markov chain has only 4 states and 5 transitions, figure 4.12

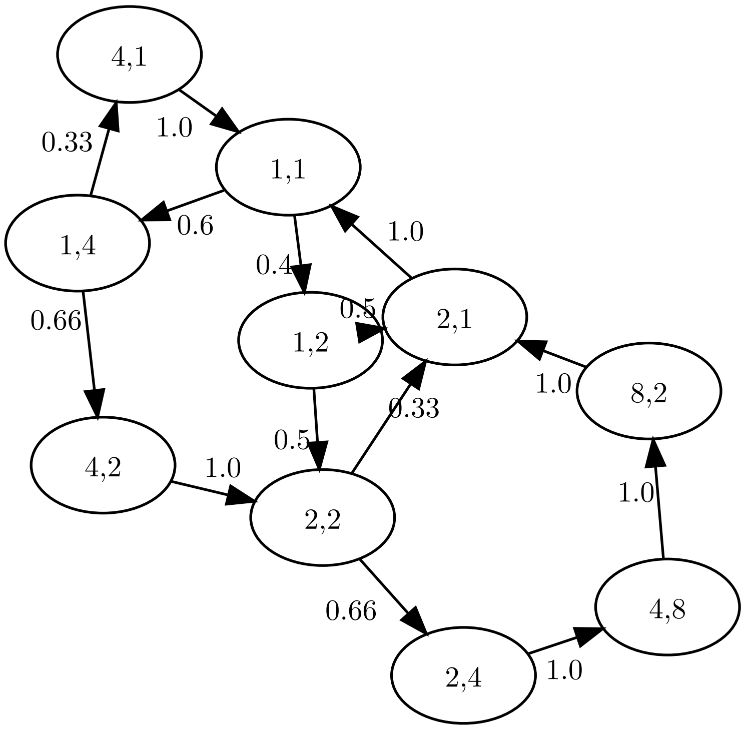


Figure 4.11: Graph of 2nd order Markov chain with event space E ={1,2,4,8}

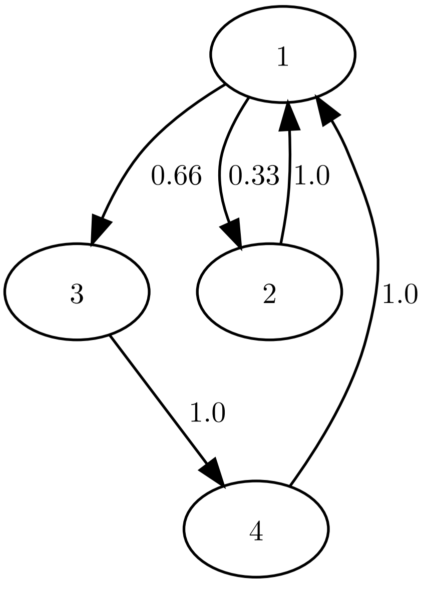


Figure 4.12: Graph of 1st-order Markov chain with event space E = $\{e_1 = (2,1,1,4), e_2 = (1,1,2,2,1,1), e_3 = (2,2,4), e_4 = (8)\}$, the number of node corresponds to index of an event

Chapter **5**

Practical part

The task of the thesis is to create a seamless transition between two songs using musical structures, music theory, and computer science. The transition is created via analysis of the songs followed by synthesis using the analyzed characteristics of the songs. This part is separated into sections: Analysis and design, Analyzer, Generator, and System evaluation.

In the first section, constraints on the songs are listed. The songs had to be preprocessed before they met the constraints. The process is described in Preprocessing. In the last section, the design of the system is described.

Section Analyzer goes through the first process of the proposed system. The analyzer describes how to acquire different attributes from songs that are then used to generate the transition. The attributes are either musical structures or Markov chains, which are used for the analysis of both melody and accompaniment.

The generator works with the attributes created in the analyzer and applies them to the synthesis of the transition. The generator creates two types of transition based on the method of melody synthesis. The final compositions are then evaluated by participants.

5.1 Analysis and design

5.1.1 Constraints in songs

To limit the number of possible implementations, we make some constraints on the input songs. A quick explanation of why and their importance is given in each of the paragraphs.

1. Single instrument

2. Separate bass from melody

3. Melody as monophony

4. Single key
5. Accidentals
6. Chord progressions
7. Time signature
8. Bounded bars
9. Single tempo
10. No rests

Single instrument - the song is limited to a single instrument as multiple instruments would add more complexity to the analysis. The instrument intended to capture the song is the piano, as it enables to simultaneously play bass and melody. The other instruments can be generated from the composition by simplification. Melody (voice leading) can be selected for single-note instruments and bass for accompaniment with various instruments such as piano or guitar.

Separate bass from melody - to be able to create a new melody and chord progression, it has bass and melody in separate channels.

Melody as monophony - the melody cannot have simultaneous notes as the analysis of the melody works only with a sequence of notes. This is not an issue as many songs are represented with voice-leading and accompaniment.

Single key - because the key signature of the song is not given and identifying the scale is part of the analysis, the song is expected to be written in a single scale.

Accidentals - accidentals are expected to not be present in the song

Chord progressions - each measure (bar) of the accompaniment contains only pitches of a single chord

Time signature - all songs can only have time signature of $\frac{4}{4}$. This is used so that extraction of the chord progression is easier. The system could be changed so that variable time signatures are allowed, but this would also mean that the chords can only change each measure.

Bounded bars - all notes cannot span more than one bar. Because in one of the approaches the song is split into bars, the notes that span more measures can become an issue. The notes could be split into two, however, that would mean that the song is played twice. For some of the notes, the mentioned method is used, but not necessarily as some were edited by adding extra notes to the melody.

Single tempo - all songs are constrained to use only one tempo. And the tempo is defined at the beginning of the song.

No rests - Rests are substituted by the repetition of the following notes or prolongation of the preceding note. Usually, songs do not have many rests within the song, if so they are quite short and the above substitution is not changing the song too much.

Summary The constraints of the song enable better manipulation of the songs and eliminate the complexity that songs can have. As an example, in figure 5.1 a small section of the song "River flows in you" by Yurima [33], is edited using software Musescore [34] and with adjusted MIDI section from EVA[29]. The issues were with chord progression inconsistency and this is fixed by doubling the tempo of the song (to 130 BPM) and all durations of the notes are also lengthened. Another problem is that the bars are not bounded (some notes are spanned in two bars), the notes are divided into two notes with each note being in a different bar.

5.1.2 Preprocessing

Before I can create the model and generate a new song, a few issues with MIDI files have to be considered. As MIDI describes only the standard for communication, there are no restrictions on how to form songs and represent them in MIDI. The variations can arise from different tools transcribing notation to MIDI. Even though the song could sound the same, in MIDI, the song could be encoded in different ways. Most of these issues originated from my implementation of loading the MIDI. In retrospect, I would have selected a different form of representation of songs as input to the system as the implementation turned to be a quite time-consuming part, which could be easily substituted.

PPQ inconsistency To understand this issue, I quickly describe how the tempo of a song works. In MIDI, the tempo of a song is affected by two values, the first is the PPQ [30] defined in the header, and the second tempo event,

[29]The loading and saving of the MIDI was edited so that the issues were removed automatically

[30]number of ticks per quarter note

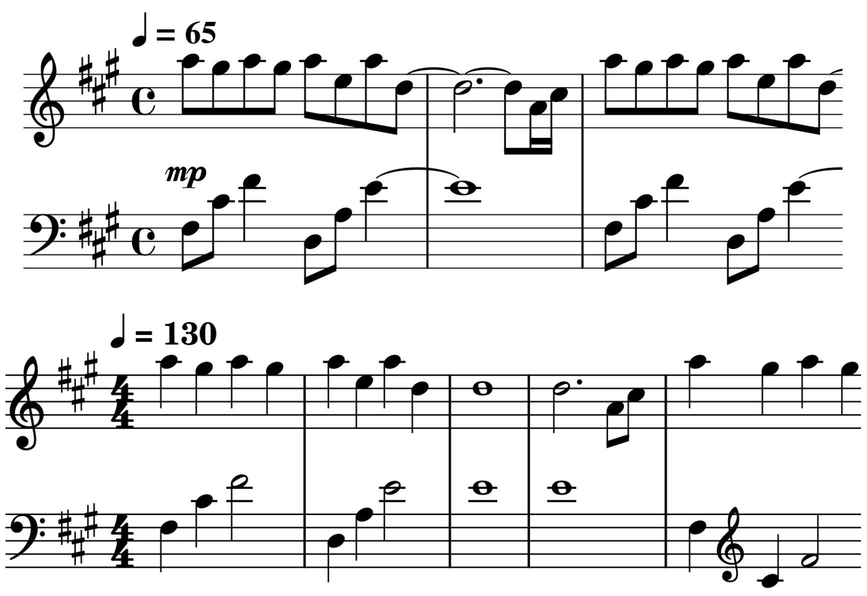


Figure 5.1: First bars of edited score of River flows in you by Yurima[35]

which sets the time in microseconds per quarter note (MPQ). The tempo event is possible to occur many times in the song, nevertheless, the songs are constrained to be only in a single tempo, so the tempo event is at the beginning of the song. Musicians use BPM (Beats per minute) to measure the tempo of songs. It describes how many quarter notes(beats) can be played in sequence in one minute. If we wish to obtain this value from MIDI, we use the following equation:

$$\text{BPM} = 60 * 1000 \ / \ \text{MPQ}$$

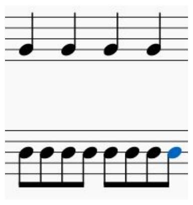


Figure 5.2: Upper part are 4 quarter notes and lower part has 8 semi notes. If the upper part would be have BPM double of the lower part they would have same tempo

The issue arises from two different values of PPQ used. Consider the same song created twice, first with PPQ: 512 and Tempo: $5 * 10^6$, and second with PPQ:256, Tempo: $5*10^6$. If combining these two songs into one, we would get twice as fast a second repetition. This means that to be able to combine two songs into one, it is necessary to have both songs in the same PPQ format.[31] In this thesis, I transformed every song to be in PPQ of 480. It is done by calculating the ratio $current_PPQ/480$ and all ticks of the events multiplied by this ratio. Just a quick example, left events with PPQ=240 → ratio=2, right events transformed to PPQ=480:

```
Tempo(0,750002)          Tempo(0,750002)
NoteOn(0, 69)            NoteOn(0, 69)
NoteOff(228, 69)         NoteOff(456, 69)
NoteOn(240, 71)          NoteOn(480, 71)
NoteOff(354, 71)         NoteOff(708, 71)
NoteOn(360, 69)          NoteOn(720, 69)
NoteOff(816, 69)         NoteOff(1632, 69)
NoteOn(640, 74)          NoteOn(1680, 74)
NoteOff(954, 74)         NoteOff(1908, 74)
```

Deviations in durations Notes in MIDI can have different lengths for the same type of duration, usually caused by the special notation of the notes. For example, these can be staccato (shortly played notes), fermata, or accent. Or variations in terms of a few ticks to fit the measure more evenly. These variations result in inconsistent durations of the same note in MIDI format. Even notes without any extra notation are still in incorrect ratio to PPQ, which is caused by the nature of the MIDI. As an example, the figure 5.3 shows a simple melody, with corresponding events on the right.

When the notes are extracted, the durations (shown in the figure 5.5) are for QN = 455, EN = 227, and HN = 911, PPQ of the melody is 480, meaning that it should be 480 pulses per QN.

If we have a look at QNs, they have 455 ticks, instead of 480 as denoted in the header. The expected representation of a song acquired from MIDI files is a sequence of notes with a limited number of durations. For this purpose, a simple solution of identifying the closest notes to the expected values could be created, creating a set of allowed durations. However, this solution would shorten the length of notes slightly and operations such as separating notes into bars would be more complicated. Besides, in some songs, for the same note duration, slight changes in tick number were present (+-5ticks). This way we would lose information about a certain type of duration used and create multiple representations instead of one.

For the next possible solution, we have to look at the start of the events. The starts of each note (NoteOn events) look like their tick distance fits the 480 PPQ frame. This way, a straightforward solution subtracting the next

[31]This is also important to do for the analysis as we need the same number of ticks for the same notes.

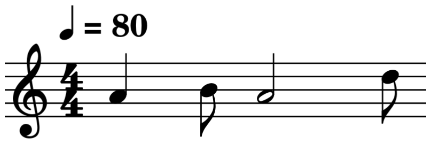


Figure 5.3: One bar as a representative made in Musescore

```
Tempo(0,750002)
NoteOn(0, 69)
NoteOff(455, 69)
Tempo(480,750000)
NoteOn(480, 71)
NoteOff(707, 71)
Tempo(720,750002)
NoteOn(720, 69)
NoteOff(1631, 69)
Tempo(1680,750002)
NoteOn(1680, 74)
NoteOff(1907, 74)
```

Figure 5.4: Midi events of simple melody

```
Note(0,69,455)
Note(480,71,227)
Note(720,69,911)
Note(1680,74,227)
```

Figure 5.5: Notes extracted from the melody in figure 5.4. Attributes of note are tick, pitch, and duration respectively.

tick of the NoteOn event from the current NoteOn could be used to create durations. For this example, the solution would work beside the last note. [32] Unfortunately, if the song is polyphonic[33] the above solution cannot be used. Within the same track, the distance to the next NoteOn event does not have to be correct, this applies also if rests are used.

The final solution is recalculating the durations to the closest duration in the list of allowed durations. This list is created from the value of PPQ and has values from $\frac{1}{32}$ duration to the whole. In the example,5.6 the list with a PPQ of 480 is created. Even though only 16th notes are allowed in the song, the thirty-second note is listed, because each note can also have a dot notation.[34] When creating a new duration for note, the note is compared to the list of allowed and the higher closest is assigned [35]. If no duration is found,

[32]solving this in the song would not be an issue as we could for example do something like mapping it to the closest duration already defined

[33]allows simultaneously played notes in accompaniment

[34]the description of the dot notation is in section Duration

[35]max deviation allowed is 10% of the note

the combination of successive notes is tried. This can be seen in the figure 5.6 which is created from notes5.5. If the tick information, the notes used for analysis are obtained.

```
Note(0,69,480)
Note(480,71,240)
Note(720,69,960)
Note(1680,74,240)
```

Figure 5.6: Notes extracted from 5.4. Attributes of note are tick, pitch, and duration respectively.

Accidentals - manipulation This problem is created after key signature analysis and when I transform the pitch of notes to the intervals of the scale. As the scale is a list of pitches of the chromatic scale, to represent notes in terms of the scale, all notes have to be from that scale. However, this does not have to happen as accidentals can occur both from falsely identifying the scale or accidentals still being present in the song. [36]

One of the solutions could be to use a 12-tone chromatic scale and express all notes in terms of this scale[37]. Or have a type of mark describing a raise in semitone for each note. This would still keep the knowledge of the scale of the song. On the other hand, we would have to remember for each note an extra attribute, which would create a more complex model. And as accidentals do not occur that often in a song, my solution is simply ignoring them.

5.1.3 Design of the system

The system, named Electronic Virutal Artistist (EVA), consists of two parts, the analyzer and generator. In the figure 5.7, a basic idea of the system is shown.

The input to the analyzer is two songs in MIDI, which are transformed into the internal representation. The analyzer identifies features of the songs and creates models representing melody and accompaniment. The first attribute identified is the key signature of the whole song, and from it scales for melody and accompaniment are created.

For the melody of the songs, two approaches are used, one uses event space, where the event is a pair of duration and pitch. The first splits the melody into a sequence of pitches and the second into a sequence of durations of length one bar. Markov chain is created for both approaches and the transition matrices are models of melody.

[36] pitches that are out of the scope of scale

[37] all MIDI values are within this scope

Accompaniment is handled separately from the melody. The accompaniment is first sequenced to bars and then for each bar, only pitch classes are considered to predict the chord type used in the bar. The rhythm patterns of accompaniment are not analyzed. From a database, a model of already identified songs using a 1st-order Markov chain is collected and then retrained with the songs. This results in a transition matrix describing the chord progression of the song.

Then generates, mainly via the transition matrices, individual sections of the transition. In the transition, 3 scales appear. The first part is generated in the scale of the first song. The middle part is in the scale created by defining a distance measure on the circle of fifths and choosing a scale that has ideally the same distance from both scales of the two input songs. The last part is again in the scale of the following song.

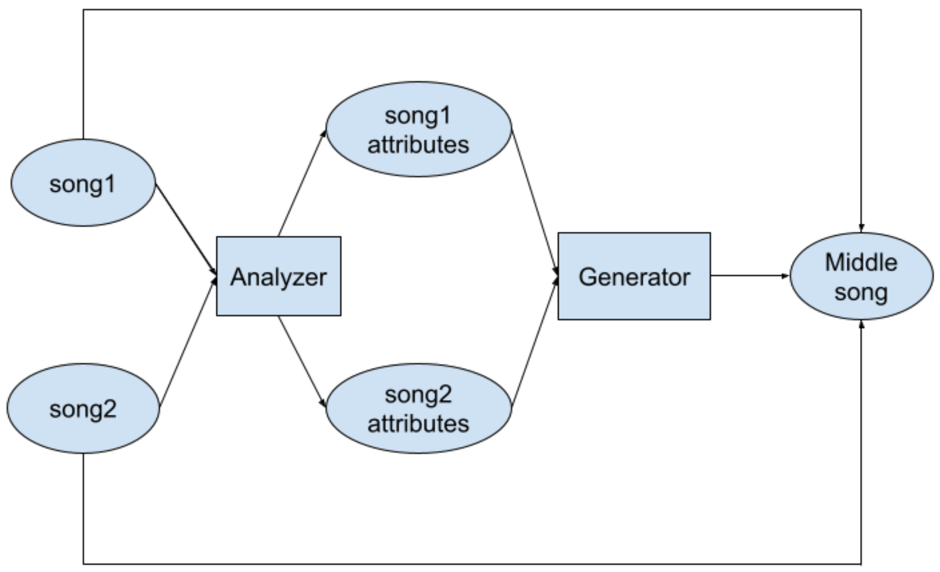


Figure 5.7: The system to create short composition connecting two songs together

5.2 Analyzer

The analysis is the main part of the EVA. It loads the songs from MIDI and transfers them to the internal representation. The songs are throughout the analysis independent of each other. Apart from some variations in transition matrices using melody pitch, the song is analyzed the same way regardless of the order it arrives in the system. The order is considered in the generator.[38]. Both songs go through the same steps, therefore, a description of a single song will be given throughout this section. The analyzer is structured in order of the acquired attributes. The attributes of the songs with their shortcuts:

- SM – Scale of Melody
- SB – Scale of Bass
- M1 – Markov chain created from pitch
- M2 – Markov chain created from the rhythm of bars
- M3 – Markov chain created from the pitch of the melody
- M4 – Markov chain representing the chord progression

5.2.1 Key signature

Before creating and analyzing the melody and bass of the songs, it is necessary to identify the key of the song as in both parts the scale is the key musical structure. The theory of the process is described in section Key Detection - Scale detection, where the pitch profile of the song correlation coefficient is calculated. The pitch profile of the song is created by taking all notes (from NoteOn events) and extracting the pitch, this creates a sequence of pitches. Each pitch is mapped by $mod12$ to its pitch class. For each pitch class, the abundance of notes having the same pitch is calculated.

To obtain the scale of the song, the correlation coefficient is calculated using major and minor profiles, and all pitch classes, in total 22 correlation coefficients (11 pitch classes times 2 types of scales). In the figure 5.8, pseudo-code of the scale detection is described. The major and minor profiles are listed in table 4.2. The highest value of the coefficients describes both the root pitch class and the scale type of the song.

To create scales of both the melody and accompaniment, the octave of the root has to be found. This is achieved by calculating the mean of all pitch values. The mean value describes the center, the first pitch that is below the mean value and is from the same root pitch class is the root. The pair root and scale type identify the scale for both melody and bass.

5.2.2 Melody pitches analysis

Because the melody in this thesis is monophonic, it enables representing it as a sequence of notes. As some characteristics of tones are omitted, the note can be represented as a pair of pitch and duration. In the MIDI, the sequence

[38]Input song1,song2 will result with a different transition than input song2, song1

```
function detect_key:
    major <- major pitch profile
    minor <- minor pitch profile
    song <- song pitch profile
    max <- describes max value from correlations,
                the scale type and root is stored too
    for (root <- range 0 to 11):
       k_minor <- calculate correlation between minor,
                            with current root
       k_major <- calculate correlation between major,
                                with current root
       if k_minor or k_major > max:
            max <- update max
    return max
```

Figure 5.8

is acquired via the NoteOn and NoteOff events. The process is described in section 5.1.2.

Definition 5.2.1 *Given $d \in D$, where D is set of durations and a $p \in \mathbb{N}, 21 \leq p \leq 108$ is pitch, an ordered pair $n = (p, d)$ is said to be a note and the set of all possible notes is defined as N*

The set of durations can vary, that is why I do not specifically define the set of durations. As said in the section 5.1.2, the values of durations are constructed from a set 32^{nd} Note,SN,EN,QN,HN,WN with relative ratio to the PPQ. Therefore, one of the possibilities would be defining it in terms of ratio or simple set $1, 2, 4, 8, 16, 32$ respectively, where each value represented would have to be transformed to MIDI ticks by multiplying it by a constant m = PPQ/8. However, in the implementation, I work with values derived directly from MIDI ticks. The exact values are not important.

To relate to the elements of the note n, the notation n.p and n.d is used to describe pitch and duration respectively.

Definition 5.2.2 *An ordered set $m = m_1, \ldots, m_n, n \in \mathbb{N} \wedge \forall i \in < 1, \ldots, n >, m_i \in N$ is said to be a monophonic melody*

From now on, I will be working with the sequences of pitch values and durations separately, the sequence of pitches of a melody m is labeled $m.p = m_1.p, \ldots, m_n.p$, and m.d for durations. For the representation of the pitches of the melody already the m.p could be used, however, that is possible with a large number of songs analyzed. Instead of using a large dataset, I chose to utilize the knowledge of the scale and express the pitch melody in terms of scale and then create a step function.

Definition 5.2.3 *Given a vector* $x = (x_0, \dots, x_i, \dots, x_n), \forall i \in [0, n), 0 \leq x_i < x_{i+1} < 12$, *and a value* $r \in N$, *where* $x \bigoplus r = (x_0 + r, x_1 + r, \dots, x_n + r)$ *using this we define scale as an ordered pair of* $s = (r, x)$ *where* r *represents root note and* x *steps within this scale,* S *is then set of all scales*

To express notes in terms of the scale, I define two functions, one is a function, which returns the value of the pitch, by indexing in the scale:

Definition 5.2.4 $\xi : S \times \mathbb{N} \to N, where \xi(s, n) = s.r + s.x(n \textbf{ mod } 12) + d * 12$, *where* $0 <= n + 12 * d <= 12$

And function abs, which takes the pitch and returns the index of the given scale

Definition 5.2.5 $abs : S \times N \to \mathbb{N}$, *where* $abs(s, n) = i$ *if* $\exists i$, *such that* $\xi(s, i) = n$

The next definition needed is the step function

Definition 5.2.6 *Given a vector* $x = (x_0, x_1, \dots, x_n), x \in \mathbb{N}$ *and a function* $\delta : x \to \mathbb{Z}, \delta(x_i) = x_i - x_{i-1}, i \in \mathbb{N}, 1 \leq i < n, \delta(x_0) = x_0$ *thus creating new vector* $\delta(x) = (x_0, x_1 - x_0, \dots x_i - x_{i-1}, \dots, x_n - x_{n-1})$ *the new vector* $\delta(x)$ *are the steps of vector* x

To obtain the steps of intervals within a scale. This can be now expressed with the above definitions as:

Definition 5.2.7 *Given* $m \in M, s \in S$, *where* $\forall x \in m, \exists i$ *such that* $\xi(s, i) = x$ *then* $abs(m.p) = (abs(p_0), \dots, abs(p_n))$ *and* $\delta(abs(m.p)) = (abs(p_0), abs(p_1) - abs(p_0), \dots, abs(p_n) - abs(p_n - 1))$ *are defined as melody scale steps,*

MSS is a vector of $\mathbb{Z}$, where range depends on the scale used and values are usually within $[-14, 14]$. At this point, the MSS is paired with the sequence of durations to describe melody.

Because this is the last part before Markov chains, an example with one bar melody from figure 5.3,[39] is transformed into scale steps of scale $s = (C5, Major)$. The description of obtaining notes is described in section Deviations in durations, thus this part is skipped. The melody m $(69, 480), (71, 240), (69, 960), (74, 240)$ and scale $(72, (0, 2, 4, 5, 7, 9, 11))$ are given. In the table 5.1, the iterative steps of obtaining MSS are listed.

[39] As a reminder these are notes (A3,QN),(B3,EN),(A3,HN),(D4,EN)

operation	**sequence**			
m	(69,480)	(71,240)	(69,960)	(74,240)
$m.p$	69	71	69	74
$abs(m.p, s)$	-2	-1	-2	1
$\delta(abs(m.p, s))$	-2	1	-1	3

Table 5.1: The process of obtaining scale steps from melody

Generating generic melody In this work, what we are trying is to create a transition from one song to the other in a way that the connection is seamless, but still, it is expected that the generated part will be different from the songs. This is partly achieved by creating MSS and also by creating a generic melody. In this part, I describe how to generate a simple melody.

I chose an approach of a simple probabilistic context-free generator inspired by the study ”Musical Composition with Stochastic Context-Free Grammars”. [13] The proposed generator for pitch used in the study is a probabilistic generator where each note is created by intervals within the scale. In the table 5.2 probabilities describe the transition as interval. These intervals apply for downward and upward intervals with a 50/50 probability. I created 3 out of 10 songs using this technique.

Interval	1st	2nd	3rd	4th	5th	6th	7th	8th
Probabilities	0.05	0.3	0.2	0.1	0.2	0.05	0.05	0.05

Table 5.2: Two possible set of probabilities based on the ”Musical Composition with Stochastic Context-Free Grammars” [13]

Proposed generator for melody The first adjustment is using a probabilistic context-free grammar describing the probability distribution that is used. The nonterminal symbols describe different features of tonality.[40] The proposed features and their symbols in the grammar are:

- small steps(S) – conjunct melodic motion
- one note with higher probability(C) – centrality
- large steps(L) – acoustic consonance

The generator adjusts the probabilities of intervals for some movements. The proposed PCFG is listed in table 5.3, the rules are after the nonterminal, and the probabilities are in the last column, where they correspond to the rules respectively. The output of this system is a sequence of terminals (0-7) and the letter ”c”. This sequence is then transformed to a given scale, where

[40]The features are described in section Tonality

```
    input: sequence of symbols, scale and central note
    output: sequence of MIDI values
function pitchesFromString:
    previous note <- central note
    pos <- 0
    for every symbol in symbols do
        if the symbol is c
            note <- find closest pitch to the previous
                        note using central note pitch
            previous note <- note
            pos <- position of note in scale (abs)
            add note to the new sequence
        else
            //it can only be number here 0-7
           pos <- (pos + symbol)
           note <- note from the scale indexed by pos
           previous note <- note
           add note to the new sequence
```

Figure 5.9: A pseudo-code that obtains pitches from string generated by proposed PCFG

0-7 describes steps in the scale. Letter c is transformed to the closest pitch of the central note to the previous note.

As an example, (c,-1,c,3) is generated from PCFG.[41], which is in C-Major scale and C4 central pitch, transformed to values (60,59,60,67), in scientific pitch notation (C4, B4, C4, G5). Below is a pseudocode for the transformation to pitch.

rules	**N = (Start,M,S,L,C)**	**T = (0,1,2,3,4,5,6,7,c,e)**	**probabilities**
S	$\rightarrow$	CM | MC | CMC	**[0.3,0.3,0.4]**
M	$\rightarrow$	SM | LM | CM | m | e	**[0.4,0.15,0.25,0.15,0.05]**
S	$\rightarrow$	SS | s | e	**[0.3,0.5,0.2]**
L	$\rightarrow$	LL | l | e	**[0.1,0.4,0.5]**
C	$\rightarrow$	CSC | c | e	**[0.25,0.05,0.4,0.3]**
m	$\rightarrow$	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	**[0.05,0.3,0.2,0.1,0.2,0.05,0.05,0.05]**
s	$\rightarrow$	0 |1 | 2 | 3 | 4	**[0.2,0.4,0.3,0.15,0.05]**
l	$\rightarrow$	2 | 3 | 5 | 7	**[0.1,0.1,0.45,0.1,0.25]**

Table 5.3: The probabilistic context-free grammar for generating melody

The table 5.4 describes one of the generated sequences from the grammar and its transformed sequence with the midi values. From my point of view,

[41]the -1 was created by 50% random chance of transforming to downward motion

Symbols from grammar	(2, 1, 8, 1, 2, 1, 8, 1, 8, 2, 4)
MIDI values	(64, 65, 60, 62, 65, 67, 72, 71, 72, 69, 76)
Scientific pitch notation	(E5, F5, C4, D5, F5, G5, C5, B5, C5, A5, E6)

Table 5.4: Example grammar output, where C-major and root C were used. The second line is created from the output by transforming the c to closest pitch to the previous note, the numbers are transformed using scale and they refer to steps within the scale(50% for upward and downward motion)

this method implements better tonality features of melodies than the simple probability generator described earlier.

With these two methods, 10 generic melodies are created. 3 with the first method and 7 with the other. The length of each melody is approximately similar to the length of the input song.[42]

Markov Chains on MSS At this point, MSS is created for the melody of the given song and also a generic melody. The event space used are the values of the MSS, typically[-14,14]. The obtained transition matrix will be a model for melody in the generator.

For the melody, I have selected a 3rd-order Markov chain to capture the dependencies in melody. In addition to the definitions and description of the Markov chains in section 4.2.4 the *training is weighted*, equivalent that the same input used w times.[43]. To get the desired Markov chains. The training has to be performed.

The pseudocode is described in figure 4.8. The train method takes as input a sequence of events and iteratively goes through the sequence and updates the transition matrix values. For both types of approaches, the training is performed twice, once with the generic melody and weight of 1, and the second time with the song's melody and weight of 5, so that the generated song is affecting the probabilistic generator more, and the melody is more similar to the input one. The output of this part is a transition matrix, that is after the training normalized, and used as input to the generator, it is labeled as M1.

5.2.3 Melody rhythm analysis

The necessary part of the melody is the rhythm pattern. To better capture the structure of the melody, the durations are grouped into bars. The sequence is created by a simple algorithm from the sequence of durations. A set D = {2,3,4,6,8,12,16,24,32} is a set of durations in the song. The rhythm pattern is defined as:

[42]The length of the melodies has to be at least 80% length of the song and at max 120%.

[43]In the implementation the input is not used w times but instead each cell of the transition matrix is incremented with value w instead of 1

Definition 5.2.8 *The sequence $d = (d_1, d_2, \ldots, d_n)$, $where d_i \in D$ is called a rhythm pattern. If $\sum_{i=1}^{n} d_i = 32$ it is said to be a bar rhythm pattern or a bar*

Simply the bar is of length one whole note, where the whole note has a value of 32. Here we fix the value of the quarter note to 8, but in the implementation, I work with the exact PPQ as it is more convenient for debugging.

The sequence of bars is used to create 1$^{\text{st}}$-order Markov chain. In these, the event space is the bars of the song. It is labeled as M2

5.2.4 Rhythm and pitches of melody combined

For the melody, one more model is created. This time MSS is used in combination with duration. The event space E, $e \in E, e = (p, d), where p \in [-14, 14] \wedge d \in D$, is used for the 2nd-order Markov chain. Similarly to melody pitch, the generic melody is used. This time the melodies are matched with melodies of the song, m.d, if the generic melody is shorter, m.d is shortened, and if the generic melody is longer, it is shortened to fit m.d. This way, 10 generic songs are created and used to train the Markov chain, the real melody is trained with the weight of 5. Normalization is performed at the end. The result is a model describing the pitch+rhythm of the melody(M3).

5.2.5 Chord progression analysis

The proposed analysis of chord progression is again done with the knowledge of the scale and use of Markov chains. A large part of the generator is created with the probabilities from In the section Chord Progression, the obtained graph from the website is described and the probabilities listed, these values are used by the generator to create new sequences. Because the Hook theory model is created using 1st-order Markov chains, in this implementation the same is used to analyze the chord progression of the song and update the obtained transition matrix.

The key part of chord progression analysis is the extraction of the chord type. What I try to accomplish here is identifying the type of the chord in terms of Roman numerals (the root of the chord), with a proposition that all chords are triads.

Representation of bass The bass is extracted differently than the melody from MIDI. Identifying the individual notes and durations is similar to the melody, however, the sequence of single notes would not represent the bass properly.

The internal representation is a sequence of bars, where each bar consists of a sequence of either notes or chords. This is extracted directly from the MIDI algorithm that takes a sequence of pairs, where the first item is the tick in the midi file and the second is a note. The individual notes are then structured based on the overlapping. As defined in the section Constraints in songs, all notes played simultaneously have to have aligned NoteOn and NoteOff events. [44] This way chords are created by combining notes that have the same tick.

In the figure 5.10, a representation of accompaniment is at two levels. One is with Roman numerals (level 1) and the second with durations accompanied with a sequence of steps (level 2). The rhythmical structure is at level 2, and in this work, it will not be implemented. The next part works with level 1 only.

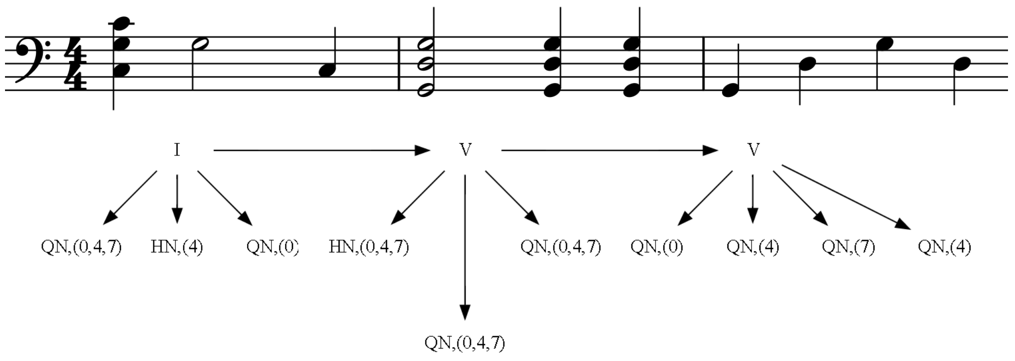


Figure 5.10: Example of desired representation of chord progression with I, V, V scheme

Chord type extraction At the start, the representation of the bass is in terms of bars with a sequence of chords (or notes), in identifying the chord type only pitches of the chords are needed, so for each bar, the pitches are extracted. Unlike the pitch profile of the song, the number of times the pitch is in the bass is omitted and the only information collected is a set of pitch classes present in the bar.

As an example, the first measure from figure 5.10 would be represented as {C,G}, the second as {G,D}, and the third {G,D}. Moreover, as the scale is known and the pitches are in terms of pitch classes, the intervals in the scale (index of scale) can be used, thus {C, G} in C-major would be {0,4}. The final transformation of the bars, before using correlation to identify the chord type, is creating a vector that represents all indexes of scale, the values of the vector are either 0 or 1 based on the profile extracted. Therefore, (0, 4) is transformed to (1,0,0,0,1,0,0). I call this "a profile of the chord".

As the songs are expected to use only major/minor triads in their accompaniment, the chords can be identified by their root. I propose a similarity measure created with the correlation. As a reminder, the triad consists of

[44] all notes start playing together and finish at the same time

```
input: chord of the bar, scale
output: root of the chord
function detect_chord:
    triad <- (10,1,5,1,7,1,2)
    chord_profile <- profile of chord
    max <- describes max value from correlations,
                the root is stored
    for (root <- range 0 to 7):
       k <- calculate correlation of chord_profile,
                and triad, with current root
       if k has higher value than max:
            max <- update max
    return max.root
```

Figure 5.11: Pseudo-code detecting type of the chord

notes formed by (0,2,4) scale intervals. The method uses the chord profile (10,1,5,1,7,1,2). It is created based on the knowledge of the significance of intervals in the triad (octave, fifth...). The next part is similar to the one used in KS, the correlation between the vectors of the chord and the triad. It is created 7 times and the highest value describes the most likely chord. The figure 5.11 describes the process of detecting the root of the chord, thus, determining the chord type.

The result of this part is a sequence of chord types, where each value describes the root key of the bar. For the example 5.10, the sequence (0,4,4)[45] is the identified chord progression.

Training the Markov model The sequence of chord types is used as the sample space to train 1st-order Markov chains. The event space is a root value (0-7). The transition matrix is created the same way as before and then combined with the transition matrix collected from Hook theory with ratio 1:1, meaning both the transition matrices are first normalized, and after that, all probabilities are added together. The resulted matrix(M4) has to be again normalized and the transition is the model describing chord progression.

5.3 Generator

The generator is the part where all models and attributes from the analysis are collected and used to synthesize a transition between two songs. These

[45]Internally the root is described in terms of index in the scale, however (0,4,4) describes (I,V,V) chord progression

attributes are a bit different for individual approaches. For the first approach, the attributes of one song are:

- SM – the scale of the melody
- M1 – a model of melody pitches
- M2 – a model of melody rhythm in bars
- SB – the scale of the bass
- M4 – a model of chord progression

For the second these are:

- SM – scale of the melody
- M3 – model of melody pitches + rhythm
- SB – scale of the bass
- M4 – model of chord progression

The attributes are the same for both songs, thus, for the first 10 attributes used. The songs in the generator are labeled as song1(S1) and song2(S2), describing the preceding and the following song.

Generating the song is separated into bass and melody generation as they are generated independently. The whole song is separated into 4 parts of 4 bar length for both the melody and bass. In the figure 5.5, description of the layout. For each part, the corresponding column describes what song attributes are used. Apart from the obvious song1, song2 attributes, the mid-scale appears. It is an intermediate scale that is created from song1 and song2 scales. The last row describes the bass of accompaniment. Since I have not created any model for the bass, the bass is created via either WN chords or arpeggios. The former describes playing a chord for the whole length of the bar, and the latter is chord[46], where individual notes of the chord are played in sequence. The format used to create arpeggio using intervals root, fifth, and octave, and notes played as root → fifth → octave → fifth. The last step is descending, the others are ascending.

Mid-scale The scale created for the middle part is created with the help of Distance and the similarity of scales. First, a path is created via traversing a lattice of scales (3.8). The scales of both songs are used as a measure and iterative steps between the two scales are taken. The algorithm in figure 5.12 describes the iterative process of creating a sequence of scales. In terms of the lattice, the scale movement is only horizontal. The mid-scale is generated once

[46]typically spanning more octaves

Part	1	2	3	4
scale	song1	mid-scale		song2
melody	song1	song1	song2	song2
progression	song1	song1	song2	song2
bass	simple WN chords	Arpegios		simple WN chords

Table 5.5: the layout of the generator, describes what scale is used at the given part and rhythm pattern is used for bass. First two parts relate to song1 models and parts 3,4 to song2.

the sequences (s1,s2) have scale equivalent scales at their ends. The mid-scale is always taken from the first sequence, and it is the last scale.

As an example, the search for mid-scale of scales C-major and B-minor. The first step is C → G, next step Bm → Em, and then the condition Em == G, therefore the G major scale is the mid-scale of scales C-major and B-minor.

```
input: scale1,scale2 <- two scales
output: mid-scale
function findMidScale:
   s1 <- sequence one
   s2 <- sequence two
   append scale1 to s1
   append scale1 to s2
   while s1.last is not scale equivalent with s2.last:
       left <- do a left horizontal step from s1.last
       right <- do a right horizontal step from s1.last
       the shorter distance to s2.last is selected and
                            appended to s1
       if the s1.scale is scale equivalent with s2.last
           break

       left <- do a left horizontal step from s2.last
       right <- do a right horizontal step from s2.last
   return s1.last
```

Figure 5.12: Mid-scale finding algorithm in pseudo-code

Generators For every part of the song, the generators are used to create different sequences. In section Probability generation, the description of how to generate sequences from probabilistic state transition systems is described. I will not go through the process again, only describe the individual steps used to combine the parts, and how to get the parts.

Chord progressions The chord progressions are simple, for every part, I generate a new chord progression of length 4 with a starting state *I*, the model used is M4. Therefore, for each bar, we obtain a sequence of 4 chord types (root values), that are then linked to the scales at the specific part. In all parts, chords are created from 3 notes: root, fifth, and octave. In the first and last measure, the chords are simply played simultaneously for WN duration. In 2nd and 3rd part, the arpeggios are applied. This way the chord progressions are created.

Melodies For the pitch of the melodies, in both approaches the same is done. As the melody was transcribed into MSS, the similar reverse procedure has to done for the generated sequence. First the sequence has to be "unstepped", as the individual numbers in this sequence represent steps, the sequence values are summed. Let a vector x = $(x_0, \ldots, x_n)$ be the sequence describing the MSS, the vector y = $(x_0, x_1 + x_0, \ldots, \sum i = 0nx_i, \ldots, \sum i = 0nx_i)$ then describes intervals in scale. To get individual pitches, the vector is applied to function ξ defined in Analyzer, $\xi(s, y) = (\xi(s, y_0), \ldots, \xi(s, y_n))$. The function takes scale and a index describing the relation in scale. As an example, the process in table 5.1 is reversed to obtaining MSS.

For each of the approaches, the appropriate attributes are used. In approach 1, M1 with M2 is utilized. For each part, the first step is generating 4 bars from M2, where the start is a random bar of the transition matrix. From the bars, the number of pitches is calculated by the summation of the durations in individual bars. The pitches are synthesized from M1 and the two generated sequences are matched together.[47] In approach 2, M3. The generated sequence is not sequenced into bars, however, I iteratively generate the sequence for the melody and once the duration of the sequence matches or exceeds the length of 4 bars, the sequence is used. If the latter happens, the last note of the sequence is shortened to match the 4 bar scheme. The sequence is split into duration and pitches(MSS), to obtain the values of pitch in terms of the corresponding scale. Finally, all parts are assembled as described in table 5.5.

5.4 Implementation of the system

The implementation was created in the programming language Scala 3 [36] in IntelliJ IDEA [37].[48] Besides the mentioned structures created, I tried to create a more complex internal structure using my knowledge of music theory, however, most of it did not come to fruition. The internal representation of the

[47]Of course the reverse MSS procedure is expected to be made

[48]Scala 3 was not officially released during the time I started working on the project, it was released in May 2021

hierarchical structure of music tries to capture all similarities of the musical notation. The figure 5.13 describes the dependencies of the individual classes.

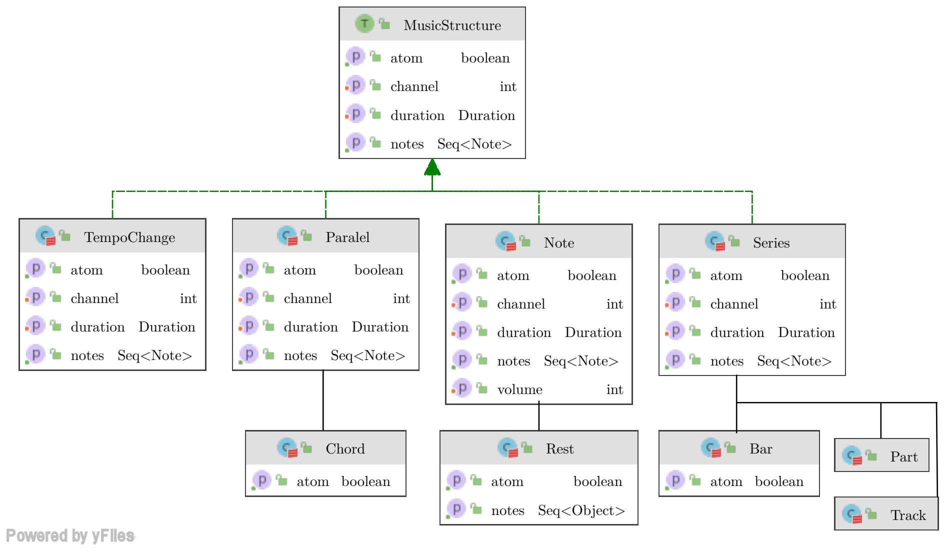


Figure 5.13: Class hierarchy used to represent song internally in the implementation from IntelliJ IDEA

5.5 System evaluation

To be able to evaluate the system, several things are needed. The songs that will be used to generate the transitions, and an evaluation method of the transition. Since music is subjective, I have chosen to evaluate the system with a third person.

Songs in evaluation The selected represent different emotions in music. The emotions are joy, sadness, and anger, which are the basic emotions identified in music [38], and hope. All songs were edited before using it in the system, as at least one of the constraints defined in section 5.1.1 violated.

The adjustments of songs are small, mainly changing the polyphonic melody to monophony and adjusting the time measure, so the chord progression changes every bar. The edited songs are also included in the content of the enclosed CD. The songs and corresponding characteristics are listed in the table 5.6.

Evaluation For the evaluation of EVA, I generated songs with the songs listed in the table 5.6. These songs are listed with their attributes in the table 5.6. The input to the system is each combination of 2 songs, creating 12

Name	Heart of Courage	River Flows In You	Summer	You’re Gonna Go Far Kid
Author	Two Steps From Hell	Yiruma	Joe Hisaishi	The Offspring
Source	[39]	[35]	[40]	[41]
Genre	Soundtrack	Pop, classical	Soundtrack, classical	Rock
Emotion	Hope	Sadness	Joy	Anger
Key sign.	B♭-minor	F♯-minor	D-major	C-major
Length	61	92	109	92
Chords	I,II,VII,II	I,VI,III,VII	VI,IV,V,I	VI,IV,I,V

Table 5.6: The list of sample songs used for the evaluation of the system

different pairs. These pairs are used twice, the first time with generator type 1 and the second with generator type 2. Thus, in total 24 generated songs.

The first evaluation was done with a participant, who was a male student(23 years old) with basic knowledge of music theory. The participant was given all 24 songs labeled with the corresponding song1 and song2 inputs to the system. He was also given 4 songs that were used as input to the system. He listened to the songs in a quiet room using headphones and was allowed to listen to the songs multiple times.

The evaluation for each song was done by listening to the transition and answering 6 questions. Each composition is scored using 5-point Likert answers.[49] The questions are listed in table 5.7.

Q1:	”Does the song sound pleasant?”
Q2:	”Does the song sound real?”
Q3:	”Does the song sound interesting?”
Q4:	”Does it retain same emotion as in input songs?”
Q5:	”Is the song similar to the input songs?”
Q6:	”Is the transition fluent?”

Table 5.7: Questions used for evaluation of the system

The results of the evaluation of approaches 1 and 2 are listed in tables 5.8, 5.9 respectively.

One more evaluation was created to test if the songs used for generation can be identified from the transition easily. The test was done with a volunteer that did not hear any generated songs before. The participant was a female (26-year-old) with intermediate knowledge of music theory, the environment

[49]Simple answers ”Strongly disagree”(1), ”Disagree”(2), ”Undecided”(3), ”Agree”(4), ”Strongly agree”(5)

TYPE	2	Q1	Q2	Q3	Q4	Q5	Q6
heart	kid	2	2	3	2	3	3
heart	river	2	1	2	4	2	2
heart	summer	2	4	4	4	2	4
kid	heart	4	4	5	1	1	5
kid	river	1	2	2	1	1	4
kid	summer	2	2	4	3	3	4
river	heart	2	4	4	2	2	4
river	kid	2	4	4	2	2	4
river	summer	2	2	4	2	2	4
summer	heart	4	5	5	5	4	4
summer	kid	2	2	4	3	2	2
summer	river	4	4	4	5	3	3
AVERAGE		2,42	3	3,75	2,84	2,25	3,59

Table 5.8: Results of the evaluation using approach 1

TYPE	1	Q1	Q2	Q3	Q4	Q5	Q6
heart	kid	2	3	2	1	3	1
heart	river	3	3	4	2	2	3
heart	summer	4	4	4	4	4	1
kid	heart	2	2	2	2	1	3
kid	river	3	4	4	4	4	4
kid	summer	3	4	4	4	4	4
river	heart	2	3	4	4	2	4
river	kid	2	2	2	4	1	1
river	summer	2	2	2	4	3	4
summer	heart	4	4	4	4	2	4
summer	kid	4	4	4	4	5	4
summer	river	4	2	3	2	3	4
AVERAGE		2,92	3,09	3,25	3,25	2,84	3,09

Table 5.9: Results of the evaluation using approach 2

was noisy, and the songs were played by a speaker. They were asked to combine individual songs so that the songs are in order song1 $\rightarrow$ generated song $\rightarrow$ song2.

The participant was given 4 songs that were used for generating before the test began to listen to them carefully. After that 6 unlabeled transitions were given[50], and the participant matched them to the original songs. The results were that 2 were guessed completely correctly, 1 completely incorrect, and the

[50]3 songs for each approach

last 3 only one of the songs correct. This suggests that the composition is not too similar to the original song, but there is still some information as more than half of the songs were guessed correctly.

Result The interesting result is that when the song was pleasant (Q1), the similarity to the previous songs was not always high (Q5), for all Q1 above 3, the average value of Q5 was 3,1.

The results suggest that the method of using both the pitches and duration together in the model creates a more pleasing song. I believe that to accomplish a more seamless transition, bass of the song should be added to the system. The middle section was crucial in combining the two songs. The use of arpeggios was a good approach as it created a richer sounding accompaniment, and the melody was not so dominant in that part.

Overall, only a few of the songs sounded unpleasant, which I consider as a big positive. Most of it is thanks to the tonality that was used to create these sequences. Upon listening to the songs reviewed as unpleasant, they had slightly dissonant intervals in them, which appeared because the chord progression was generated independently of the melody. This could be avoided if the generated song was automatically analyzed again and all dissonant intervals would be substituted by more consonant ones.

5.6 Suggested improvements

My work leaves a lot of space for improvement as some of the parts of the song were not utilized at all. The most important part would be the rhythmical pattern of accompaniment.

For the analysis, I suggest splitting the song into sections: beginning, middle, final, and each analyze separately. These sections would be then be used in similar parts in the generation of the transition song. In addition, finding repetitions of larger structures, such as chorus and verse might be interesting. The transition could then be generated by removing the outro of the song and creating a transition with the use of bridges and choruses.

One of the obvious improvements could be implementing parts that were removed using constraints or ones that were ignored, such as the volume or timbre.

Post-processing part, where the transition would be edited so that slightly dissonant intervals are removed. Besides that, reediting the transition to better describe the tonality features could also extend this work.

Conclusion

In this work, I have identified different approaches, that are used for the algorithmic composition. Based on that I chose to use a knowledge base from music theory and probabilistic models.

The system created compositions based on analysis of the song and collecting models for different parts of the song. The generation was created for two types of approaches. The approaches were evaluated with a participant, that listened to 24 generated songs and answered several questions describing the transitions. The results of the evaluation suggest, more pleasing composition arises, when the pitch and rhythm are used together, however, the transition was more fluent, when rhythm was formed in bars.

The goal was to create a song that would seamlessly connect two input songs. The results suggested that the transition does retain some qualities of the songs, but altogether the quality of the composition does not perform better than human. The transitions could be used as an underlying composition for composers.

Bibliography

[1] Herremans, D.; Chuan, C.-H.; et al. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, volume 50, no. 5, 2017: pp. 1–30.

[2] Fernández, J. D.; Vico, F. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, volume 48, 2013: pp. 513–582.

[3] Hiller, L.; Isaacson, L. Illiac Suite. Score. 1957.

[4] Kirnberger, J. P. *Der allezeit fertige Menuetten- und Polonaisencomponist.* George Ludewig Winter, 1767.

[5] Lerdahl, F.; Jackendoff, R. *A generative theory of tonal music.* The MIT Press, 2017.

[6] Hamanaka, M.; Hirata, K.; et al. Musical structural analysis database based on GTTM. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, ISMIR 2014, 2014, pp. 325–330.

[7] Hamanaka, M.; Hirata, K.; et al. Implementing "A Generative Theory of Tonal Music". *Journal of New Music Research*, volume 35, no. 4, 2006: p. 249–277, doi:10.1080/09298210701563238.

[8] Quick, D.; Hudak, P. Grammar-based automated music composition in Haskell. *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & amp; design - FARM '13*, 2013, doi:10.1145/2505341.2505345.

[9] Holtzman, S. R. Using Generative Grammars for Music Composition. *Computer Music Journal*, volume 5, no. 1, 1981: p. 51, doi:10.2307/3679694.

[10] Jurish, B. Music as a formal language. *Photographs: Uwe Vollmann and Irmgard Jäger Photo selection: Reni Hofmüller Typesetting: michon, Hofheim Printing: Fuldaer Verlagsanstalt*, 2004: p. 81.

[11] Lopes, H. B.; Freitas, A. Probabilistic (k, l)-Context-Sensitive Grammar Inference with Gibbs Sampling Applied to Chord Sequences. In *ICAART (2)*, 2021, pp. 572–579.

[12] Worth, P.; Stepney, S. Growing music: musical interpretations of L-systems. In *Workshops on Applications of Evolutionary Computation*, Springer, 2005, pp. 545–550.

[13] Perchy, S.; Sarria, G. Musical composition with stochastic context-free grammars. In *8th Mexican International Conference on Artificial Intelligence (MICAI 2009)*, 2009, pp. 120–130.

[14] Kong, Q.; Li, B.; et al. GiantMIDI-Piano: A large-scale MIDI dataset for classical piano music. *arXiv preprint arXiv:2010.07061*, 2020.

[15] Gillick, J.; Roberts, A.; et al. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning*, PMLR, 2019, pp. 2269–2279.

[16] Pachet, F.; Roy, P.; et al. Finite-length Markov processes with constraints. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 2186–2293.

[17] Carlton, D. I analyzed the chords of 1300 popular songs for patterns. This is what I found. [online], Nov 2019. Available from: `https://www.hooktheory.com/blog/i-analyzed-the-chords-of-1300-popular-songs-for-patterns-this-is-what-i-found/`

[18] van den Oord, A.; Dieleman, S. WaveNet: A Generative Model for Raw Audio. 2016. Available from: `https://deepmind.com/blog/article/wavenet-generative-model-raw-audio`

[19] Barreau, P. [online], 2016. Available from: `https://www.aiva.ai/`

[20] Dong, H.-W.; Hsiao, W.-Y.; et al. Audio and Speech Processing (eess.AS). Sep 2017.

[21] Dhariwal, P.; Jun, H.; et al. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[22] Source, O. Magenta. 2018. Available from: `https://magenta.tensorflow.org/`

[23] Moffat, D. C.; Kelly, M. An investigation into people's bias against computational creativity in music composition. *Assessment*, volume 13, no. 11, 2006.

[24] Wolfe, J. [online], 2005. Available from: `https://newt.phys.unsw.edu.au/jw/notes.html`

[25] Commons, W. File:Harmonic Series.png — Wikimedia Commons, the free media repository. [online], 2020, accessed 19-June-2021. Available from: `https://commons.wikimedia.org/w/index.php?title=File:Harmonic_Series.png&oldid=476090038`

[26] Roel, H. Roel's World Blog " Music Blog " Music & Geometry. [online], Aug 2020. Available from: `https://roelhollander.eu/en/blog-music/music-geometry/`

[27] Commons, W. File:Overtone.jpg — Wikimedia Commons, the free media repository. [online], 2020, accessed 10-June-2021. Available from: `https://commons.wikimedia.org/w/index.php?title=File:Overtone.jpg&oldid=456239889`

[28] Tymoczko, D. *Geometry of Music Harmony and Counterpoint in the Extended Common Practice.* Oxford University Press, 2014.

[29] Paiement, J.-F. Probabilistic models for music. Technical report, EPFL, 2008.

[30] Raphael, C.; Stoddard, J. Harmonic analysis with probabilistic graphical models. *https://archives.ismir.net/ismir2003/paper/000032.pdf*, 2003.

[31] Krumhansl, C. L. *Cognitive foundations of musical pitch.* Oxford University Press, 2001.

[32] Musescore. Arrival of the birds - The Cinematic Orchestra. [online], Jan 2021. Available from: `https://musescore.com/guestinpiano/arrival-of-the-birds-the-cinematic-orchestra`

[33] Langevin, E. River Flows In You by Yurima. [online], May 2021. Available from: `https://musescore.com/user/12461571/scores/3291706`

[34] Musescore. [software], Dec 2018. Available from: `https://musescore.org/en/3.0`

[35] Musescore. River Flows In You. [online], May 2021. Available from: `https://musescore.com/user/12461571/scores/3291706`

[36] Odersky, M. [software], May 2021. Available from: `http://dotty.epfl.ch/`

[37] JetBrains. IntelliJ IDEA Ultimate. [software]. Available from: `https://www.jetbrains.com/idea/`

[38] Fritz, T.; Jentschke, S.; et al. Universal Recognition of Three Basic Emotions in Music. *Current Biology*, volume 19, no. 7, 2009: pp. 573–576, ISSN 0960-9822, doi:https://doi.org/10.1016/j.cub.2009.02.058. Available from: `https://www.sciencedirect.com/science/article/pii/S0960982209008136`

[39] Musescore. Heart of Courage - Two Steps from Hell. Jan 2020. Available from: `https://musescore.com/user/14912166/scores/5004943`

[40] Musescore. You're Gonna Go Far Kid. [online], Jan 2020. Available from: `https://musescore.com/user/4463561/scores/1547436`

[41] Musescore. Summer (Joe Hisaishi). [online], Mar 2021. Available from: `https://musescore.com/sunbinamra/scores/429546`

Appendix **A**

Acronyms

AIVA Artificial Intelligence Virtual Artist.

ANN Artificial Neural Network.

BPM Beats Per Minute.

CA Cellular Automata.

CFG Context Free Grammar.

EA Evolutionary algorithm.

EVA Electronic Virutal Artistist.

GTTM Generative Theory of Tonal Music.

KBS Knowledge based system.

MIDI Musical Instrumental Digital Interface.

MM Markov Model.

MPQ Microseconds Per Quarter note.

MSS Monophony Scale Steps.

PCFG Probabilistic Context-Free Grammar.

PPQ Pulses Per Quarter note.

RNG Random Number Generator.

STS State Transition System.

Appendix **B**

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
├── src ....................................... the directory of source codes
│   ├── scala ............................. implementation sources in scala
│   ├── python ........................... implementation sources in python
│   └── thesis .............. the directory of LaTeX source codes of the thesis
├── text ............................................ the thesis text directory
│   └── thesis.pdf .......................... the thesis text in PDF format
└── gen ............................ the directory of songs used in evaluation
    ├── input ................................... the directory of input songs
    ├── approach1 ............ the directory of songs created with approach 1
    └── approach2 ............ the directory of songs created with approach 2
```