Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# Music Harmony Analysis:

## Towards a Harmonic Complexity of Musical Pieces

Master's Thesis

Bc. Ladislav Maršík, 2013

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# Music Harmony Analysis:

## Towards a Harmonic Complexity of Musical Pieces

Master's Thesis

| | |
|---|---|
| Course of study: | Informatics |
| Branch of study: | 2508 Informatics |
| Department: | Department of Computer Science |
| | Faculty of Mathematics, Physics and Informatics |
| | Comenius University in Bratislava |
| Supervisor: | Mgr. Martin Ilčík |
| | ICGA TU Wien |
| Date and place of publication: | June 2013, Bratislava |

Bc. Ladislav Maršík

76586059

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# THESIS ASSIGNMENT

| | |
|---|---|
| **Name and Surname:** | Bc. Ladislav Maršík |
| **Study programme:** | Computer Science (Single degree study, master II. deg., full time form) |
| **Field of Study:** | 9.2.1. Computer Science, Informatics |
| **Type of Thesis:** | Diploma Thesis |
| **Language of Thesis:** | English |

**Title:** Music Harmony Analysis: Towards a Harmonic Complexity of Musical Pieces

**Aim:** Harmony analysis of musical pieces based on tonal harmony. Finding out the harmonic complexity of musical pieces. Creating an application for harmony analysis.

| | |
|---|---|
| **Supervisor:** | Mgr. Martin Ilčík |
| **Department:** | FMFI.KI - Department of Computer Science |
| **Head of department:** | doc. RNDr. Daniel Olejár, PhD. |
| **Assigned:** | 18.10.2011 |
| **Approved:** | 24.10.2011 |

prof. RNDr. Branislav Rovan, PhD.
Guarantor of Study Programme

.......................................... Student

.......................................... Supervisor

76586059

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Ladislav Maršík
**Študijný program:** informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
**Študijný odbor:** 9.2.1. informatika
**Typ záverečnej práce:** diplomová
**Jazyk záverečnej práce:** anglický

**Názov:** Harmonická analýza: Smerujúc k harmonickej zložitosti hudobných diel

**Cieľ:** Analýza harmónie hudobných diel založená na tonálnej harmónii. Nájdenie harmonickej zložitosti hudobných diel. Vytvorenie aplikácie pre harmonickú analýzu.

**Vedúci:** Mgr. Martin Ilčík
**Katedra:** FMFI.KI - Katedra informatiky
**Vedúci katedry:** doc. RNDr. Daniel Olejár, PhD.

**Dátum zadania:** 18.10.2011

**Dátum schválenia:** 24.10.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.......................................................
študent

.......................................................
vedúci práce

**Declaration**

I hereby declare, that I wrote this thesis by myself, under the guidance of my supervisor and with the help of the referenced literature.

............................................

**Acknowledgments**

I would like to thank my supervisor, Martin Ilčík for the most valuable time spent with this work. His experienced ideas gave the work a real value. I thank professor Stanislav Hochel from The Bratislava Conservatory because he gave this work the foundation, building my musical knowledge. I also thank the professors from University Bordeaux 1, Pierre Hanna and Matthias Robine, for giving this work the right direction, when it needed most.

My dearest thank goes to my family. My parents and siblings are the greatest support and my grandma the greatest motivation that I could have, for my studies. I would also like to thank Pavla Liptáková, for giving me the belief and the vision, while working on this thesis.

# Abstract


| | |
|---|---|
| Author: | Bc. Ladislav Maršík |
| Title: | Music Harmony Analysis |
| Subtitle: | Towards a Harmonic Complexity of Musical Pieces |
| University: | Comenius University in Bratislava |
| Faculty: | Faculty of mathematics, physics and informatics |
| Department: | Department of computer science |
| Supervisor: | Mgr. Martin Ilčík<br>ICGA TU Wien |
| Date and place of publication: | June 2013, Bratislava |



In this work we present a new theoretical model for finding out the complexity of harmonic movements in a musical piece. We first define, what the yet undefined term, harmonic complexity, means for us, finding different perspectives. Our basic model is based on tonal harmony. Utilizing the fundamental rules used in western music we define a grammar based model in which transition complexity between two harmonies can be evaluated as the computational time complexities of derivation from one harmony to the other. In graph representation the transition complexity can be found as the shortest path between the two harmonies. For these purposes we have created an object oriented model that implements the theoretical model. In the end we deploy the system, Harmanal, capable of analyzing harmony transitions from MIDI and WAV input. We have used Harmanal for comparing the overall harmony complexities of musical pieces from different genres. Moreover, we find Harmanal as a new possibility for enhancing music information retrieval tasks such as implementing a recommender system for music.



**Keywords:** harmonic complexity, music complexity, harmony analysis, chord transcription, chord progression, music information retrieval, recommender system

# Abstrakt

| | |
|---|---|
| Autor: | Bc. Ladislav Maršík |
| Názov práce: | Harmonická analýza |
| Podnázov: | Smerujúc k harmonickej zložitosti hudobných diel |
| Škola: | Univerzita Komenského v Bratislave |
| Fakulta | Fakulta matematiky, fyziky a informatiky |
| Katedra | Katedra informatiky |
| Školiteľ: | Mgr. Martin Ilčík |
| | ICGA TU Wien |
| Dátum a miesto vydania: | Jún 2013, Bratislava |

V práci uvádzame nový teoretický model pre nájdenie zložitosti harmonických prechodov v hudobnom diele. Najskôr popíšeme, čo doposiaľ nedefinovaný pojem, harmonická zložitosť, pre nás znamená, pričom uvažujeme viaceré možné perspektívy. Náš základný model stavia na tonálnej harmónii. Extrahovaním fundamentálnych zákonov používaných v teórii západnej hudby sme skonštruovali model založený na formálnych gramatikách, v ktorom možno harmonický prechod medzi dvoma harmóniami zhodnotiť ako časovú zložitosť odvodenia z jednej harmónie do druhej. V reprezentácii na grafe môže byť zložitosť prechodu nájdená ako najkratšia cesta medzi harmóniami. Pre tieto účely sme vytvorili objektovo orientovaný model, ktorý implementuje popísaný teoretický model. Nakoniec predstavujeme systém Harmanal, schopný analyzovať harmonické prechody získané zo vstupov MIDI alebo WAV. Systém Harmanal sme použili na porovnanie celkovej harmonickej zložitosti hudobných skladieb z rôznych žánrov. Navyše, systém Harmanal považujeme za novú alternatívu pre zefektívnenie úloh týkajúcich sa práce s hudbou na počítačoch, ako napríklad vyhľadávanie doporučenej hudby pre používateľa.



**Kľúčové slová:** harmonická zložitosť, hudobná zložitosť, harmonická analýza, transkripcia akordov, akordický rad, vyhľadávanie hudby, odporúčanie hudby

# Foreword

Back in the days when I was studying music composition, the biggest questions I have had on my mind were – how to make the music more interesting? How to create more memorable tunes? Will the listener find the same aspects of music beautiful that I do? If you were ever creating some sort of art, you might have ended up with questions like these… Similarly, if you have your favorite music pieces, what does really *make* them your favorite?

I have found, that along with the personal preference of everyone of us, it is also the function of our musical experience and knowledge. If we have devoted ourselves into studying music harmony or music itself, our preference changes. We would eventually recognize the patterns of compositions and find the differences between simple and more complex music. Interestingly enough, sometimes the more we *know* about the possibilities in music, the more we can incline towards simpler music. More often, however, we may get tired of the established practices and seek different, more complex progress. In the result, the skilled composer of the 21st century can create music that may sound too complex or perhaps too minimalistic and thus not beautiful for an inexperienced listener.

Generally speaking, it is difficult to decide whether simpler music can be more popular, or vice versa. It is a subjective matter. But what we can conclude is, that introducing a term music complexity can be helpful. Intuitively, our personal preference of music should correlate with our preferred *complexity* of music. And for the music, such complexity can be measured.

Well, can it be measured? That is more of a musicologist's question. I would always prefer a thorough analysis by a knowledgeable music analyst over an analysis made by a machine, in the same way that I would prefer human-made art over a machine-made product. But, given that even musicology does not have any general rules for finding out the complexity, and not many works were yet done in the mathematic or informatic field too, I decided to make the new pathways. The result will prove itself good if it can be used by both, musicologists and software developers.

As strong as I believe that computers can not supersede the position of human in producing and analyzing music, I also believe that music and mathematics vastly overlap, if not, are the same. In that fashion I started to use different applications easing the work of a musician, like notation softwares or music sequencers. Later I started creating my own. First of them, Ear training application [13] with chord naming model I will reference in this work, too. The next one you are reading right now. And, more are yet to come.

If you find this work useful for any kind of expansion or you are interested in further discussion, please contact me at: *laci@marsik.sk*.

# Table of Contents

# 1 Introduction

## 1.1 Music harmony

*„The most important in music is its harmony.“*

Ilja Zeljenka, Slovak music composer

A great music has several qualities. It takes melody to make us memorize and hum the music on the street. It takes good rhythm to make us dance on the music at the discotheque. For popular songs, lyrics and a good chorus can relate us even more to the song. And then there is music harmony, tones sounding together, that creates the atmosphere and the depth of music. What should we use to analyze the true complexity of music?

Studying the music more and more, it is the harmony and its changing that gives us the best platform for analysis. Even the melody by itself can have an implied harmony, harmony that could accompany it based on its tone material. Moreover, it has been ever since the late Baroque until now that majority of music obeys certain harmony rules. That broadens our musical pieces space and gives us a way to compare pieces even from different genres and periods, using music harmony. Taking harmony as the subject of our research is therefore understandable. And throughout the work we will trust our motto by Ilja Zeljenka, because it gives us confidence that we have chosen the right aspect[1].

### 1.1.1 Definition

According to Laborecký [8], music harmony is defined as follow:

**Music harmony** is the study about the character of simultaneously sounding pitches, their meaning, transitions, functional relationships and usage in the musi-

[1] Supplementary to the theory of harmony, there is a comprehensive theory of counterpoint describing how we can combine multiple voices together. There is much more to take into account before we cast all music in the same mold and we should keep that in mind.

cal piece. It studies horizontal (subsequent) relationships in the time and vertical (concurrent) relationships among the tone space.

In other words, music harmony works with entities that represent simultaneously sounding pitches. It has them, with the help of music theory, precisely labeled and each entity has some meaning. Even more importantly, it specifies the rules that can connect these entities to the sequences. We thus obtain music, or more precisely, a musical accompaniment. There is a counterpart to harmony, which is **melody**, that floats on the top of musical accompaniment and comprises solely of sequence of tones and rests. For our analysis, we may choose to extract melody from musical accompaniment or let the melody and the accompaniment sound together.

Note that, music harmony, as we defined it, is a scientific discipline, whereas we will be interested in *the harmony of a musical piece*. Geared towards a single piece of music, we define:

**Harmony of a musical piece** is the use of simultaneously sounding pitches and chords, their character, meaning, transitions and functional relations in a musical piece.

In this work we might also use the term „harmony“ to refer to the entities (simultaneously sounding pitches) that the music harmony works with, i.e. interchangeably with the terms: chord, interval, cluster or chord with added dissonance (see chapter 2 for definitions). We realize that it may become ambiguous at times, but we hope that the positive reader will successfully distinguish all the different uses and misuses of the term.

### 1.1.2 History and tonal harmony

The music harmony has grown over the ages. If we focus on western music, starting in the late Baroque in 18th century, a harmonic thinking has originated, that we now know as functional tonality, or **tonal harmony**. Its core is that every part of a musical piece belongs to some major or minor key. It came to its very peak in music Romanticism in 19th century. After that, many composers have founded new approaches to music, moving outside the keys and breaching the tonal harmony rules. Special rules also apply to modal folk songs, jazz or polyphonic pieces. Nevertheless, rules of tonal harmony still apply to vast majority of music today and it is commonly being used as a way of teaching the basics of harmony. We will describe the aspects of tonal harmony important for this work, in the section 2.3.

## 1.2 Harmonic complexity

*„Two impulses struggle with each other within man: the demand for repetition of pleasant stimuli, and the opposing desire for variety, for change, for a new stimulus"*

Arnold Schönberg, Austrian composer and music theorist

The purpose of this section is to make the first steps to describe the harmonic complexity and also to describe how it relates to the beauty in music, which will help us realize the major motivations for this work. Now, we may all relate to, that if the music is „all the same" it may soon loose our interest. While listening, we need variety, change and the new stimulus in the coming seconds. But if we get only different harmonies, we will certainly neglect something that we can relate to, therefore we need repetition of our favorite passage, a pleasant stimuli. According to Zanette [24], these are the two fundamental principles that cast the musical form and that we expect in music. (And is it not the same in any other area of life?)

Intuitively, we may describe the **music complexity** as *the variety, the change and the occurrence of the new stimulus in music* – the more unexpected changes occur, the more is the musical piece complex. Such description has a nice consequence – as Arnold Schönberg helps us realize, the complexity should be the exact half of what we need in music. We should also take into consideration, that *random* and disordered changing of music harmonies should hardly qualify as complex (Zanette [24]). However, in general it is difficult to find out what was the composer's intention to make particular harmonic movement. We will therefore follow-up with our intuitive definition of complexity as the *variety* and *change*, but instead of giving an exact definition, we describe a model for evaluating the complexity in chapter 5. We also believe that such model can get us closer to music beauty. In the next sections we go deeper to find out what are the different approaches for building such model, and how it can help us in the real world tasks.

### 1.2.1 Beauty and complexity

Just like „The Beauty and The Beast", it is clear that the beauty and the complexity of music are two different terms. But following the fairytale storyline, we may get to the point where they find the way towards each other.

### 1.2.2 How can the complexity find its way to beauty – motivations

Inevitably, music beauty is subjective for every listener, whereas the complexity, since we seek to describe it by a general model, is not. But like we said in the foreword, every listener also has a subjective look on what is complex and what is not. In other words, we may still use something that has to do with the listener's **preferred complexity of music**. That is, the complexity of music that he or she is used to, that he or she likes. If such complexity exists, we can measure it. But then we can use such measurings to find other music that he or she will like, too.

This idea is well known as *recommender systems*, that famous internet radios or portals such as Pandora, or Last.fm[2], are using. Such systems have various implementations, filtering music based on its content, or based on other users' preferences (collaborative approach). The latter is the most frequently used approach.

[2] http://www.last.fm; http://www.pandora.com

We may also conclude, that *if* the recommending is based on music content, it is usually on the genre of the piece or the artist, which may not provide enough flexibility. Recommendation based on the music complexity is a new approach and can enhance the state-of-the-art techniques.

To begin with, we should find out the complexity of the specified genre of music, or the concrete artists. If we have good results, we may use harmonic complexity to specify the genre or similar artists more accurately, or more interestingly, to find slight differences amongst the genre. It is quite obvious that two rock bands, let's say Queen and Led Zeppelin, would have different music styles. We may end up finding that they have different complexities, too. That can be another evidence that using harmonic complexity for music retrieval is a good practice.

Similar researches were already done, finding out that usually the band or the composer uses certain „harmonic language" (e.g. The Harmonic Language of The Beatles by KG Johansson [6]). But we have not found any works done on comparing these languages. According to these works, chances are, that if we define our complexity well, we can gather such comparisons.

To summarize, we have found ourselves couple of **motivations** for this work. We would wish to create a mathematical model capable of:

1. Evaluating the harmonic complexity of the musical piece, so we can make one step closer towards, generally undefined, music complexity.
2. Finding out the harmonic complexity of music from different music periods, genres and artists.
3. Finding out the complexity of music library of the user so that it will be possible to implement a recommender system searching for the music with the same complexity – the music he or she would like.

The first motivation is filling the gap in the musicology-related terms. Interestingly enough, there are not any attempts known to us to evaluate the music complexity. However, there are works on tonal tension, voice leading, chord recognition, dissonances, and more, outputting different visualizations. It is only the music complexity that has always had the label of „subjective“ and „undefined“. The most common practice to call some music „simpler“ or „more complex“ than other was through some written or spoken analysis. Even if it was taken into consideration in some works, it was suppressed because the final product was to obtain another output such as chord sequence or visualization. Perhaps the reason why is the lack of clues in the harmony literature, where all the rules are found, but seldom they are somehow ranked or evaluated. We use the same rules, but we extract the evaluation from them, too.

In our work, in addition to building the model, we also put the second motivation into practice and we gather results interesting from the musicology perspective in chapter 7. The last motivation we leave open for future implementation, but it nevertheless remains one of our „ultimate motivations“– and it also shows how the complexity can find its way to the subjective beauty.

### 1.2.3 How can the beauty help define the complexity – approaches

Similarly to the fairytale, the beauty can help the complexity (the beast), to find its real self. Looking for the approaches to define the complexity, there is an analogy with looking for the ways to define the beauty. Imagine that we look for the most beautiful human in the world. Rather like the prince traveling the world, looking for the most beautiful princess, he may take one of these, three approaches:

1. Take all of his human anatomy books with him, along with a measuring tape, and then measuring all potential princesses and comparing his results with the books.

2. Take several friends with him, meeting the young women in the kingdom

and then at the evening campfire everyone would share their feelings about the girls they have met. He would, then, choose the girl with the best rating.

3. Have the king call out, that every young woman should get to the courtyard, forming a line. He would, then, find about the beauty of the girls by going from the first and comparing each one with the ones that he had already seen. By the end of the line he would have a good eye on how the princess should look like.

These three simple approaches represent: evaluation based on theory, evaluation based on perception and evaluation based on machine learning. All three are possible and indeed great ways, to evaluate the complexity too.

1. Music theory and the part of it, tonal harmony, describes the set of rules that, if used well, can help us to evaluate the complexity.

2. Music perception is an important and vital part of the cognitive sciences. We may get the complexity by studying the opinions or the mental processes of music listeners.

3. Machine learning is a common technique for music analysis. Teaching the program on a sample of musical pieces, using hidden Markov models (HMMs) to learn what are the expected harmony transitions, can get us to relevant results too.

Comparing all of these approaches would be a nice study, however, out of the scope of this work. We should choose one. Machine learning is a common approach, even giving the best known results for naming the harmonies, although, we might be concerned that it always has better results if taught on music from a specific genre, and used on that same genre. There is also a belief presented by De Haas et al. [3], that *„certain musical segments can only be annotated when musical knowledge not exhibited in the data is taken into account as well“*. Music

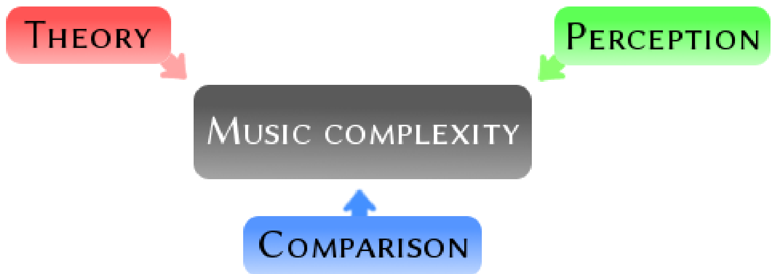



Figure 1: Approaches to music complexity

perception is a discipline on its own and lot of statistical data need to be examined to gather the results.

But having the good theoretical model first seems to be a good headstart for any future research. Thus, we have chosen the music theory, and its subset, tonal harmony as the basis for our work. We firmly believe, that, even if some other parts of music theory may enhance our results (such as theory of counterpoint or modal harmony), the way we use the key and scale based principles of tonal harmony is flexible for future modularity and apply to the majority of music we hear today, and is at the same time consistent with the related works on music theory too.

## 1.3 Goals

We here set up the main objectives of this work:

1. Create a good mathematical model for harmonic complexity based on tonal harmony
2. Create an application capable of complexity analysis
3. Compare music from different periods, genres and artists

The importance of creating a mathematical model we have already discussed and we find it a good innovation in the field of musicology and music information retrieval. Another important part of this thesis is creating an application for the end user, capable of music analysis. There is not clearly defined, who may the user of such an application be. Either a musicologist retrieving information from musical pieces, or a musician interested in chordal analysis, extracting the chords from music in order to reproduce them, or a composer playing with new harmonies, or a programmer implementing a plugin using the complexity model. Therefore, we tweak our application to provide all of these services:

- Processing WAV input for recorded musical pieces
- Parsing MIDI input for pluggable MIDI instruments
- Parsing text input for convenience
- Displaying analysis results for the whole musical piece, as well as for each harmony transitions in the piece
- On-demand analysis for input harmonies
- As a by-product to obtain complexity, we will get to analyze every single harmony from the input. Displaying the name for these harmonies can be a great help for musicians as well as theorists trying to understand how the complexity was generated

## 1.4 Outline

In chapter 2 we introduce the reader to the basic concepts of tonal harmony, understandable also for a non-musicians. The reader can find there the main definitions in order to understand, how our model works.

In chapter 3 we switch our focus for a moment and we summarize the works most related to ours. The reader can use that chapter in order to find out where trends are about now, in harmony analysis.

In chapter 4 we give the outline of Harmanal system, choosing the best fundamental techniques for our analysis from chapter 3.

In chapter 5 we introduce the basic model for harmonic complexity. The reader should not skip that chapter because it shows the main idea of this work.

In chapter 6 we describe the Harmanal application and give more insight on its components. The reader can see the application in the enclosed screenshots.

In chapter 7 we perform the analysis on music samples. The reader can find interesting results, such as – comparison of rock bands with the classical composers, or finding out which songs deviate from the majority of songs made by bands Queen or Beatles.

In chapter 8 called Future works, we take one step back and conclude our work by describing four other categories for harmonic complexity to give the picture on how the overall complexity should look like.

In the conclusion we summarize the main results of this work.

# 2 Understanding tonal harmony

In this walkthrough on tonal harmony, we will narrow our focus on definitions for those terms, that will be repeatedly used in this work. The aim is to provide clear meanings for the terms that will be used frequently, especially because around the world the terms and sometimes also the meanings differ. Another aim is to invite a non-musician reader into discussion. The musicians may, on the other side, find some interesting insights into the broad topic of tonal harmony.

The definitions were compiled from Arnold Schönberg's Theory of Harmony [21], the works of Zika and Kořínek [25] or Pospíšil [17] designed for Slovak music conservatories and a terminological dictionary by Riemann [18] and Laborecký [8]. In these works you can also find much more detailed elaboration.

**Tonal harmony** is a musical system, in which:

1. Every part of a musical piece belongs to a major or minor key.
2. Every harmony has some, close or distant relationship to the center of the key, the first degree.

We have used some terms, that, to a non-musician, might need more clarification. We will define them in the subsequent sessions.

Firstly, we quickly clarify the umbrella terms, not to confuse the readers anymore, when using terms like *music theory, musicology, music harmony*, etc. Secondly, we will hierarchically build the entities that we will work with. And lastly, we will get deeper into tonal harmony, describing the basic rules that are needed for our analysis.

## 2.1 Musicology disciplines

**Musicology** is the scholarly study of music. It is the top umbrella term that includes all musically relevant disciplines. It is just as science, as for example mathematics or informatics, but is considered social science because it studies the art creations of mankind [15]. However, moving on, we find that splitting up musicology we get on one side *historic musicology* and *ethnomusicology* and on the other *systematic musicology*, where the second mentioned contains plenty of subdisciplines that usually have interdisciplinary character.

The most important, for us, is the small, but fast growing discipline, **music information retrieval** (MIR). Its common theme is retrieving information from music, and it has many real-world applications, such as recommender systems, track separation, music retrieval by queries, or automated music transcription. Our work falls under MIR.

We were already talking about *music cognition*, which is another musicology discipline, partially falling under systematic musicology.

Other discipline right in between musicology and physics, is called **music acoustics**. It goes deep to describe how the physics in music works. But, importantly for us, there is another part of systematic musicology, that builds on the results of music acoustics, called **music theory**.

*Music theory is an applied discipline, which is, as proposed by many researchers, an applied mathematics. Although music acoustics gave the theory its building blocks, tones on the scale, and more and more evidences are there when mathematic theories have helped develop the new harmonies, such as theory of mathematic inversion, there is still some uncertainty in how much mathematics can describe music. Perhaps the reason why is that historically, music and mathematics have developed separately, one originated as an art with no axiomatic foundations, other as science. However, recent researchers are now filling the*

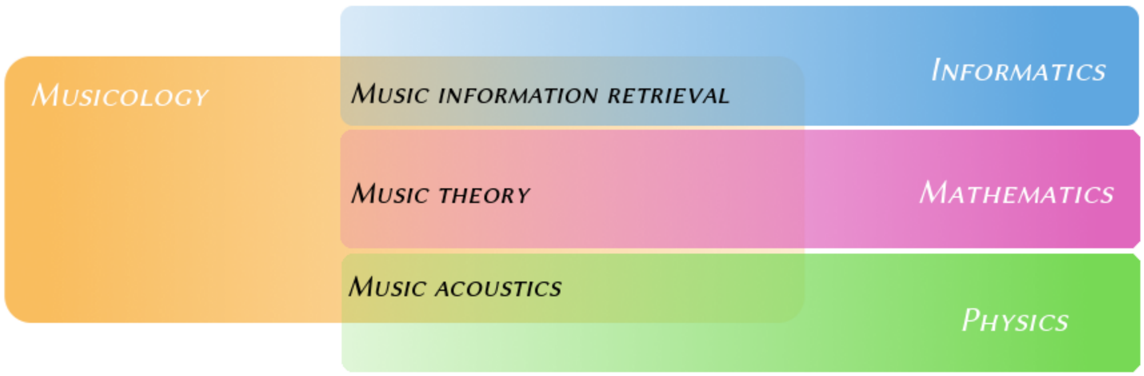



Figure 2: Musicology disciplines diagram

*gaps building new mathematical models and works*[3]*, in the same fashion as ours, to show, that the fundamental rules in music, on the top of which the mastery of the composers is built, can be described by mathematics.*

Note that, if we want to build a good new mathematic model for music complexity, we have to build it purely from the rules of music acoustics and music theory. Otherwise (using other subjective, or „artistic", reasoning), we would deviate from music theory and would not show how mathematics helps describing music. The resulting model would be wrong, just as unproved experiments cannot lead to proved theorems in mathematics. Music acoustics and music theory are bound together well, and any attempt to add a new model on a top of them, should obey these bounds and make the new model tightly related to both of them. We need to get the foundations from music theory and use the mathematic language to stay on the right track.

Then, music theory comprises of studies such as: **music harmony**, theory of counterpoint, study of musical forms, and others. Having already defined the

[3]Amongst many works we may highlight the works of David Lewin [11] [12] and Neo-Riemannian theory.

music harmony, we may conclude this overview by summarizing, that **tonal harmony** is only one concrete system in music harmony. There are others, such as modal system, using the scales commonly appearing in folk music. In the 20th century, multiple new systems arose, such as bitonality, polytonality, extended tonality or also dodecaphony introduced by Arnold Schönberg.

## 2.2 Basics of music theory

Music acoustics has helped the music theory define these basic terms:

**Tone** is an acoustic sound, that is created by regular vibration of a source.

Music theory also defines the tone as the smallest element of a musical piece, characterized by its *pitch, intensity, timbre* and *duration*. Pitch can be quantified as frequency, but it takes comparison of a complex music sound to a pure tone with sinusoidal waveform to determine the actual pitch, therefore the pitch should be considered as a subjective attribute of sound.

### 2.2.1 Finding the basic tones

From the spectrum of all audible pitches, the western music only uses a narrow set with frequencies in such distribution, that their differences may be clearly recognized by an ear (88 tones of today's piano keyboard). In this set, the two pitches, one with a double of frequency of the other, blend in the sound while played simultaneously so they resemble one sound, although they have different pitches. To these pitches, a distance of one *octave* is assigned. Within an octave, we differentiate a scale of 7 tones that is periodically repeated. These tones were assigned the alphabet letters, forming the basis of **musical alphabet**:

$$a, b, c, d, e, f, g$$

However, with stabilizing the tone $c$ as the beginning of what became a *major*

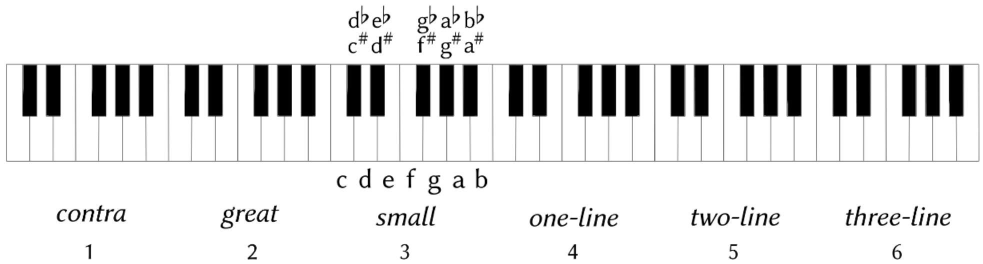



Figure 3: Tones arranged in the octaves

*scale*, we more often refer to the tone order: $c, d, e, f, g, a, b$.

To distinguish the different octaves, the labeling was established. In *Helmholtz notation* commonly used by musicians, we label the octaves from the middle and up: „one-line" ($c'$) , „two-line" ($c''$), „three-line" ($c'''$) and from the middle down: „small" ($c$), „great" ($C$) and „contra" ($C$,). Some authors prefer the *scientific notation*, simply labeling the octaves chronologically: $1,2,3,4,5,6$[4].

According to Schönberg, we can explain the basic pitches of a major scale as having been found through imitation of nature. A musical sound is a composite made up of series of tones sounding together, the **overtones**, forming the **harmonic series**. It is due to the existence of additional oscillation nodes and partial waves along with the original oscillation. The frequency of the original wave is called the **fundamental frequency** or *first harmonics* and represents the **fundamental tone** in the composite, whereas the higher frequencies are referred to as the overtones or *higher harmonics* (2nd, 3rd, ...). From a fundamental $C$, the higher harmonics are:

$$c, g, c', e', g', b\flat', c'', d'', e'', f'', g'', etc.$$

[4]On the standard piano, tones are ranging from *sub contra a* (*A*0) to *five-line c* (*C*8), MIDI tones range even from *double sub contra c* (*C-1*) to *six-line g* (*G*9).

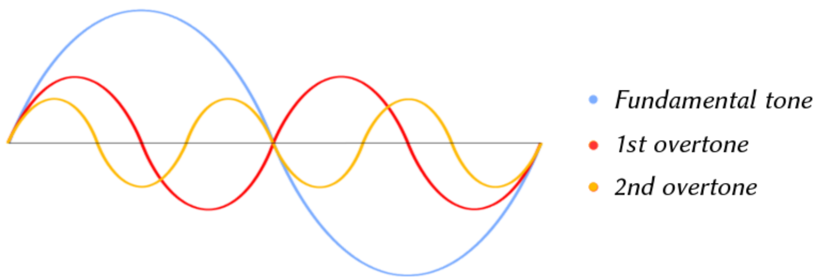



Figure 4: Harmonic series explained

The tones that occur first in the series, have also stronger presence in the composite[5]. For the fundamental tone $c$ it is therefore $g$ as the second most important component, and as such, our ear represents as a harmony when these two tones sound together. Similar assumption can also be made about the next tone appearing in the series, tone $e$. Consequently, for the tone $G$ the higher harmonics are $g, d', g', b', d''$, etc. and therefore we may conclude $g$ and $d$ as another harmony. Taking the tone $c$ as the midpoint, we should also consider the other direction (as one of the concepts of the *theory of harmonic inversion*. We have $c$ as the first overtone in the harmonic series of $f$. Following these guidelines, the 7 tones of the major scale are found.

### 2.2.2 Intervals

**Interval** is the frequency ratio of two pitches, the simplest relationship between two tones in music. From the practical perspective it can be considered as the *distance* between the two pitches, that can be derived either from their sounds or from their notation.

The harmonic series will help us locate the most important intervals that will later create the basic harmonies. Since the harmonics are from the acoustic view

[5] In fact, the actual presence of the harmonics depend on the musical instrument being played, and therefore translates to the timbre of the tone.

stationary waves with increasing number of oscillation nodes, we derive the ratio between the second and first frequency as exactly 2 : 1, the ratio between the third and second is 3 : 2, the ratio between fourth and third is 4 : 3, etc.

The frequencies ratio 2 : 1 is denoted as the **perfect octave**. It can be found for example as the distance between $c'$ and $c''$.

The frequencies ratio 3 : 2 is denoted as the **perfect fifth**. It can be found for example as the distance between $c'$ and $g'$.

The frequencies ratio 4 : 3 is denoted as the **perfect fourth**. It can be found for example as the distance between $c'$ and $f'$.

The frequencies ratio 5 : 4 is denoted as the **major third**. It can be found for example as the distance between $c'$ and $e'$.

The frequencies ratio 6 : 5 is denoted as the **minor third**. It can be found for example as the distance between $c'$ and $e\flat'$.

Following the ratios between the overtones, we have stepped out of the set of tones of a basic scale, discovering the tone $e\flat'$ in between $d$ and $e$. The difference between the major third and the minor third is the frequencies ratio 25 : 24 (we can get it by dividing the intervals). Similarly, we discover that the difference between the perfect fourth and major third is 16 : 15. These ratios, almost indistinguishable by an ear, along with couple of others occurring between the basic tones, have been denoted as the **semitone** or the **minor second**. The semitone sets the smallest commonly used distance between the tones in western music and can be used to measure the distance of larger intervals. Similarly, the distance that approximates as the double-semitone distance is denoted as the **whole tone** or the **major second**, most commonly appearing as the frequencies ratio 9 : 8.

Thus, multiple tones out of the basic major scale were added (black tones on the piano keyboard), and we differentiate two ways how to describe their presence – by two types of **accidentals**:

- If the tone can be described as created by augmenting the original tone by a semitone, we mark it with the accidental ♯ next to the original tone, and call it „**sharp**“ (*c sharp*: *c*♯, *d sharp*: *d*♯, *f sharp*: *f*♯, *g sharp*: *g*♯, and *a sharp*: *a*♯).
- If the tone can be described as created by diminishing the original tone by a semitone, we mark it with the accidental ♭ next to the original tone, and call it „**flat**“ (*d flat*: *d*♭, *e flat*: *e*♭, *g flat*: *g*♭, *a flat*: *a*♭ an *b flat*: *b*♭).

In today's music theory, the ambiguity between the different semitones in the tone scale have become impractical for some instruments. Therefore, a common interval ratio for the semitone was established, with the value of $\sqrt[12]{2} : 1$. This tuning is known as **tempered tuning**, as opposed to **just tuning** based on the exact ratios from the harmonic series.

For summary, all the commonly used intervals can be found in the table 1.

Note that, augmenting or diminishing these basic intervals using accidentals we get theoretical **augmented** or **diminished intervals** that share the same name as the original interval, but sound like a different interval, e.g. augmented third = perfect fourth.

#### 2.2.3 Scales

**Scale** is a series of increasing or decreasing pitches bounded by an octave.

**Diatonic scales** are the scales created by semitone *and* whole tone intervals. They contain 8 tones.

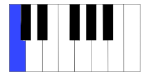
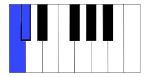
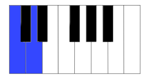
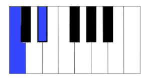
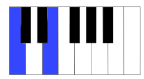
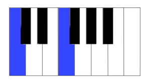
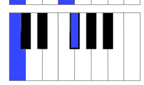
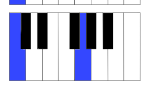
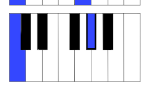
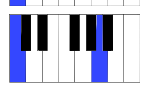
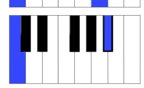
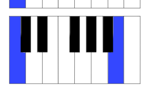
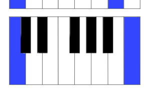
| semitones | name | picture |
|---|---|---|
| 0 | perfect unisone |  |
| 1 | minor second |  |
| 2 | major second |  |
| 3 | minor third |  |
| 4 | major third |  |
| 5 | perfect fourth |  |
| 6 | tritone |  |
| 7 | perfect fifth |  |
| 8 | minor sixth |  |
| 9 | major sixth |  |
| 10 | minor seventh |  |
| 11 | major seventh |  |
| 12 | perfect octave |  |

Table 1: Basic intervals

We divide 2 types of diatonic scales:

- **Major scales** are the diatonic scales characterized by the presence of major thirds. The most common major scale is *C major* from the basic tones we've already discussed. From tone *c*: *c d e f g a b c*. Major scales are commonly assigned a „joyful“ character.

- **Minor scales** are the diatonic scales characterized by the presence of minor thirds. The most common minor scale *a minor* is also formed from the basic tones, but from the tone *a*: $a\ b\ c\ d\ e\ f\ g\ a$. Minor scales are commonly assigned a „sad“ character.

The scales are named based on the first tone of the scale („*C major*“, „*a minor*“). The convention says, that the major scales should be labeled by a capital letter, whereas the minor scales by a non-capital letter.

The index of a certain tone in the scale is called the **degree** of the scale and is denoted by a roman numeral (I., II., ...). We may also refer to a tone using its interval from the first degree, which yields a simple expressions: *the fourth tone*, *the fifth tone*, etc.

We provide the comparison of the major and minor scales from the tone *c* in the table 2:

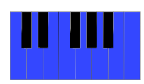
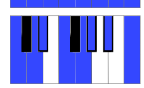
| **tones** | **scale** | **picture** |
|---|---|---|
| $c\ d\ e\ f\ g\ a\ b\ c$ | major scale |  |
| $c\ d\ e\flat\ f\ g\ a\flat\ b\flat\ c$ | minor scale |  |

Table 2: Diatonic scales

### 2.2.4 Chords

**Chord** is a set of tones with the minimum of three tones, having the intervals in between them big enough, so they may sound together without the feel of excessive density. One of these tones has to have the quality of a chord root for the chord.

**Chord root** is the tone upon which the chord can be built by stacking thirds intervals. If the root of the chord is indeed the bottom tone of the chord, we say that chord is in a **root position**. We can also obtain the **chord inversions**, by reorganizing the tones in such manner, that the root of the chord is put to the top of the chord – **first inversion** – or as the second from the top – **second inversion** – etc.

We will use the term **chord tone** for each of the tones within the tone material of the chord in the context. The term **non-chord tone** will denote a tone out of the tone material of the chord. Note that, the *tone material* implies considering the tones *mapped* to one scale, i.e. taking the tone *c* as a tone chord if the *c* from any octave is present in the chord. We will always distinguish whether we consider the real pitches where order of the tones matters, or mapped tones, the so called, **pitch classes**. In general, it is desirable to consider the real pitches for harmony study and therefore distinguish different inversions of the chord.

**Triad** is the chord in the root position made up of three tones: the root tone, the third tone and the fifth tone. It represents the harmony of a tone with its closest overtones.

Depending on the diatonic scale we use for the triad tones, we will get the two basic triads, shown in table 3.

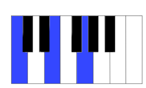
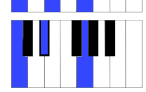
| **structure** | **chord** | **picture** |
|---|---|---|
| major third, perfect fifth | major triad |  |
| minor third, perfect fifth | minor triad |  |

Table 3: Basic triads

Applying inversions to a triad we get the three basic forms of a chord made up of three tones, shown in table 4 (on a major triad).

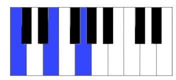
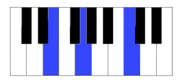
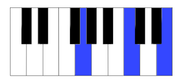
| **type** | **name** | **picture** |
|---|---|---|
| root position | triad |  |
| 1st inversion | sixth chord |  |
| 2nd inversion | four-six chord |  |

Table 4: Triad inversions

Besides major and minor triads we also distinguish **diminished triad** (minor third, diminished fifth) and **augmented triad** (major third, augmented fifth).

### 2.2.5 Basics of music notation

It is out of the scope of this work to go through all the rules of music notation. We will only briefly show how the basics work, so non-musician readers may navigate through the music samples we will use later in the work. For our purposes it will be sufficient only to localize what tones are present in the notation.

The **staff** consists of five lines. We mark the tones on the staff using the special markings, **notes**. Higher pitches are marked higher in the staff, either on the line or in between the lines. To determine the actual pitch, we need to identify at least one position on the staff, which is done by the **clef**. For the instruments with high range, the *G clef* is used, determining the $g'$ (and also derived from the letter „g“, although it resembles the letter just remotely). For specifying the pitch, an accidental ($\sharp, \flat$) may be used before the note. All the other attributes of the tone (length, intensity, the instrument playing the tone) can be derived using the special markings and guidelines – for more information we refer to Pospíšil [17].

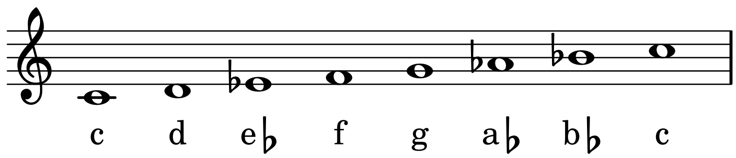



Figure 5: Notation of c minor scale

For illustration, we show the notation of the *c minor* scale in the figure 5.

## 2.3 Basics of tonal harmony

With the basic definitions, we may now proceed to the concepts of tonal harmony. First of all, the diatonic scales play a significant role in music composition, by providing the tone material that one can use to create a musical piece. Generally, they define a widespread relationship systems, called *keys*.

**Key** is the relationship system based on a major or minor scale. There are three basic levels of relationships, that define the key:

1. The series of tones: the major or minor scale.
2. The set of chords designed to build harmonies: the triads built on every degree of the scale, made out of the tones of the scale.
3. The basic chord series also called **harmonic cadence**, that sets apart triads on the first, fourth and fifth degree of the scale and gives them a role of **main harmonic functions**.

The keys are simply named by the scale they are based on, e.g. key *C* major, key *a* minor, etc. We may refer to the different tones of the key the same way, as in scales (i.e. by the degree or by the interval), and we allocate a special term for the first degree: **tonic**, the base of the key.

The complicated definition of the key simply means, that in tonal harmony, we recognize different keys, of which each defines a set of chords and their possible sequences – we may say, a set of *rules*. That of course makes creating the music or musical accompaniment much easier.

### 2.3.1 Basic harmonic functions

According to the definition of the key, some of the chords have more important roles in the key than others. They set the three basic levels of tension towards the tonic, and are the basis of tonal harmony. We recognize them as the three **main harmonic functions**:

- **Tonic** is the triad on the first degree of the key. It is the function of a harmonic steadiness and release. All the harmonic impulses origin in tonic and return back to tonic. We label it with $T$.
- **Subdominant** is the triad on the fourth degree of the key. It brings the deviation from the tonic, and is the intermediary function in between tonic and dominant. We label it with $S$.
- **Dominant** is the triad on the fifth degree of the key. It represents the maximal tension, that requires an ease, transition to tonic. We label it with $D$.

The harmonies in music start usually in the tonic. Optionally, the harmony deviates from tonic by transition to subdominant. Finally, the harmonic movement culminates in dominant and goes back to tonic again. We call this the **basic harmonic progression T – S – D – T**. According to Zika an Kořínek, it is the skeleton of every music motion in musical pieces in the tonal harmony system.

### 2.3.2 Diatonic functions

It is possible to build a triad on every degree of a diatonic scale, using the tones of that scale. Every such triad we can then assign a **function**.

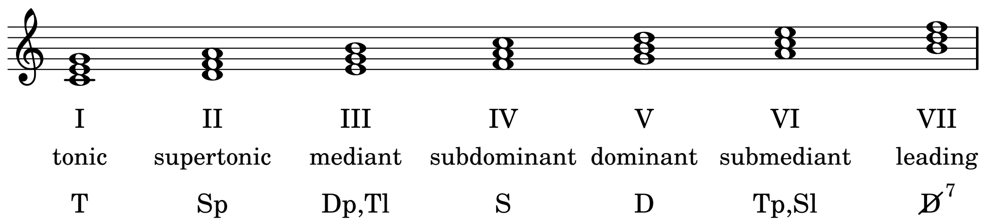



Figure 6: Diatonic functions

Note that, a function here means a certain *role* that the triad plays towards the root of the key, the tonic. We have already discussed, that the three main roles (three main functions) are: tonic, subdominant and dominant, built on (I., IV. and V. degree accordingly.

The triads on the other degrees we perceive as variants, or parallels of the three main functions. They share characteristic tones with the main functions, and are therefore capable of substituting them in certain cases.

Some theories assign each of the seven triads a function on its own – as is commonly taught in USA – whereas the others will assign the name with the respect to the main function – German approach. We can nevertheless call the triad with the name of the degree it is built on, simply I, II, III, . . . , VII (some theories would use lower-case roman numerals if the triad is minor). All of these namings are summarized in the figure 6.

According to Hugo Riemann, the originator of „functional“ approach to tonal harmony [19], we may describe the function variants in two ways. Either we get them from the main function by extending their fifth by a whole tone – thus obtaining their **parallel**, labeled with *p*, or by diminishing their root by a semitone – thus obtaining so-called **counter parallel**, labeled with *l*, since the process is also called *leading-tone exchange*.

- The triad II represents the variant of subdominant, **subdominant parallel**. We label it with *Sp*.
- The triad III represents both the **dominant parallel** and **tonic counter parallel**. We label it with *Dp* or *Tl*.
- The triad VI represents both the **tonic parallel** and **subdominant counter parallel**. We label it with *Tp* or *Sl*.
- The triad VII is an exception, although it resembles **dominant counter parallel**, the root is diminished one more semitone down, thus obtaining not a major nor minor chord, but a „diminished“ chord. However, because of its characteristic structure – upper 3 tones of *dominant seventh chord*, that we will discuss later – it is simply called **incomplete dominant seventh** and labeled $\not{D}^7$.

The study of tonal harmony also describes additional chord structures that may be considered as one of these functions – some of them will be mentioned in the next section – but mostly describes different ways of connecting these functions in music. The common transitions are: *T* – *S*, *T* – *D*, *S* – *T*, *S* – *D*, *D* – *T*, only the transition *D* – *S* is not used. According to Zika, however, we know some exceptions, e.g. when subdominant substitutes dominant only temporarily. The main message still remains though, that with simple *T* – *S* – *D* – *T* transitions, the music would be too narrow and limited and – considered simple. But the point is, that instead of main functions we may always use the variant, parallel, of the main function. This makes the music much more interesting, changing, and complex.

## 2.4 Additional definitions

In this section we will define all the rest of the musical terms, that will be used in this work. If You have enough of definitions for now, feel free to continue reading the chapter 3 and use the rest of this chapter as a dictionary that You can refer back to.

(a)

| **C** | **C♯** | **D** | **D♯** | **E** | **F** | **F♯** | **G** | **G♯** | **A** | **A♯** | **B** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.74 | 0.00 | 0.10 | 0.00 | 0.53 | 0.04 | 0.00 | 0.66 | 0.00 | 0.15 | 0.00 | 0.10 |

(b)

| **C** | **C♯** | **D** | **D♯** | **E** | **F** | **F♯** | **G** | **G♯** | **A** | **A♯** | **B** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Figure 7: Sample chroma vector of *C major* harmony (a) and a pitch class vector representation of the *C major* triad (b)

- **Alteration.** Chromatic raising or lowering of a note of a major or minor chord in order to obtain different harmony. Alteration is considered a chromatic phenomenon in the diatonic system [18].

- **Chroma.** Same as *semitone*. The term is used in different ways – **chromatic scales** are the scales that run through the twelve semitones of equal temperament. Thus, **chromatic**, as opposed to diatonic, refers to the structures or movements derived from this scale, e.g. a process of augmenting or diminishing one or more tones by a semitone, that can not be described diatonically [18]. Music information retrieval uses the term **chromas** as the synonym for **pitch class profiles**, the 12-dimensional vectors of floats, used to map the presence of the tones in the point of time in a musical piece to the 12 tones of the chromatic scale. These chromas are usually compared to chord in the pitch class representation (12-dimensional binary vectors) to approximate the chord in MIR, see figure 7.

- **Chord with an added dissonance.** First used by Czech composer Leoš Janáček, we denote the consonant chord enriched with a non-chord tone –

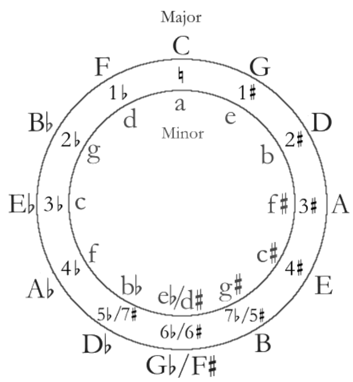



Figure 8: Circle of fifths

thus creating dissonances – as a *chord with an added dissonance*[6]. They represent the intermediate stage in between the chords and clusters. We do not use this term if the chord can be described otherwise, e.g. as a seventh chord [23].

- **Circle of fifths.** The rotation through the twelve tones of the *chromatic scale*, by fifth intervals, represented graphically in a circle. It is commonly used to represent the keys, because, starting from *C major* and *a minor*, the keys following by fifth intervals in one direction, have increasing number of tones with *sharp* accidentals, and starting from the same (*C major*, *a minor*) going the other direction, have increasing number of tones with *flat* accidentals (see figure 8). This is also a common way to determine how many augmented or diminished tones are there in the particular key – finding out the number of steps in the circle of fifths. The set of accidentals for a particular key is referred to as the *key signature*.

- **Cluster.** Cluster or **tone cluster** refers to a set of tones sounding together, with at least three adjacent tones (with a whole tone interval or smaller),

[6] In Czech language the term is more simple and has the meaning similar to *densed chord*. We prefer using the formal translation not to confuse with new terms.

where the functional substance of the chord can not be identified anymore [8].

- **Consonance.** The harmonious sound or coalescence of two or more tones, giving the impression of harmonic stability to the listener. On the basic interval scale, all of the perfect intervals, major and minor thirds and major and minor sixths are consonant [8].

- **Dissonance.** The inharmonious sound, opposite to *consonance*, that requires harmonic transition. On the basic interval scale, major second and minor seventh are considered „mild“ dissonances, whereas minor second and major seventh are considered „sharp“ dissonances. A special dissonance is also assigned to specific inharmonious sound of the tritone interval [8].

- **Leading tone.** A tone leading to another causing the another tone to be expected in harmony after the presence of the leading tone. This is usually due to a *dissonance* in the preceding harmony, that needs to be relaxed – turned into consonance. A leading tone is always a semitone down or up from the expected tone. We find leading tones especially in the diatonic scale, a semitone below the tonic (*b* leading to *c* in *C* major). But there is another type of leading tones – every sharp or flat accidental which raises or lowers the tone of the diatonic triad in the process of *alteration* introduces the tone which produces the effect of leading tone [18].

- **Modulation.** Passing from one key to another in a musical piece, a change of tonality [18]. It is used to add change to the musical piece or to highlight or create the structure of the piece. From a simplified perspective, it can be either **diatonic**, if all of the transitions can be described functionally, or **chromatic**, if the chromatic transition was used. In case of *diatonic modulation* we look for a *common chord* , called **pivot chord**, that has functional meaning in both of the keys [25].

- **Seventh chord.** The chord in the root position made of the root, the third, the fifth and the seventh (stacking three thirds on the top of each other). The seventh chords and their inversions (**five-six chord**, **three-four chord**, **second chord**), although containing a *dissonance*, are very important structures in tonal harmony. We name the seventh chord (and its inversion) based on the name of the lower triad and the name of the seventh, e.g. *major/minor seventh chord*. The importance of seventh chords lies in the fact, that for each key, **characteristic dissonances** can be found, that may, too, substitute the main harmony functions. These are: **dominant seventh chord** as a major/minor seventh chord on the V. degree, having a strong dominant character, **half-diminished seventh chord** as a diminished/minor seventh chord on the II. degree, having a subdominant character, and **diminished seventh chord** as a diminished/diminished seventh chord on the VII. degree, having a dominant character and because of its specific structure common for multiple keys, used for modulations. It is mainly the presence of additional *leading tones* that yields the usage of these dissonances in functional harmony [25].

# 3 Related works

In this chapter we provide the summary of the works most related to ours. Music Information Retrieval is a modern discipline. Before 2000 the works were scattered, focusing on different aspects of computer music. But the revolution of music distribution and storage has ignited the interest of musicians and scientists to MIR and brought to the beginning of the conferences ISMIR[7] (2000) and the yearly evaluation for systems and algorithms MIREX[8] (2005), where many more works can be found.

It is clear that our task consists of more smaller steps. Since tonal harmony provides us with rules to build chord transitions, we ultimately want to extract chords from the audio. Our final list of tasks then looks like this:

1. Extracting the features from the audio
2. Chords recognition
3. Creating a model for harmonic complexity
4. Comparing music from different music periods and genres

For each step, multiple works have been already done. In following sections we provide a quick summary of the state-of-the-art approaches, so that we can choose the best practices for our analysis in chapter 4. We also discuss, what we neglect in the previously proposed models and set the expectations for the rest of the work.

## 3.1 Extracting audio features

We are interested in obtaining the **chroma features** from the audio. The extraction is based on discrete-time Fourier transform (DFT) that takes time-domain

[7] http://www.ismir.net
[8] http://www.music-ir.org/mirex/wiki/MIREX_HOME

input and provides us with frequency-domain output. To obtain semitone-spaced chromas one must first apply transcription that takes the harmonic series of each tone into consideration and derives the approximation on what tones are sounding together. Finally, the obtained tones are mapped into 12-dimensional arrays. This algorithm has some known implementations already.

### 3.1.1 Vamp plugins

The popular implementation is the use of Vamp plugins[9]. The NNLS Chroma Vamp plugin[10] developed by Matthias Mauch from Queen Mary University of London outputs the chromas for given WAV audio. In his work [14], Mauch describes how the algorithm for solving non-negative least squares (NNLS) can be used to obtain the tones from the frequency-based data. NNLS Chroma plugin is free to obtain and re-use under GPL licence.

Another feature we might want to obtain from the audio, if possible, is the exact start and end time of the chords in the musical piece. However, the **chord boundaries** are loose, moving them in one direction or another will result in different, but possibly valid chord recognition. Some researches use various **segmentation** techniques, where the final boundaries of the chords are found as the best scoring option after matching the segments to chord templates. This approach was used by Pardo and Birmingham [16] and we explain it a little more in the next section.

Other researchers use an approach, where the segmentation is derived from a different aspect: rhythm. Chord boundaries are approximated at the time of the beats. The core idea of this method is, that the harmonic changes often appear at the beats – not only in popular music, but also in classical pieces. Conveniently enough, there is another Vamp plugin called Bar and Beat Tracker by Davies and

[9]http://www.vamp-plugins.org
[10]http://isophonics.net/nnls-chroma

Stark [22], that estimates the position of metrical beats within the music.

For the simplicity, we have decided to utilize both Vamp plugins (NNLS Chroma and Bar and Beat Tracker) for our first practical complexity analysis results. Whereas NNLS Chroma seems to be the best option, finding chord boundaries by beat tracking may introduce some inaccuracies, so it can be later changed in favor of the further musical analysis.

## 3.2 Chords transcription

The process of obtaining chords from the audio input is called **chord transcription**. Fujishima was the first to use the pattern matching method to choose from chord candidates, in 1999 [4]. From 2008, chord transcription became the common benchmarking topic at the MIREX challenge – between 7 to 19 algorithms are presented annualy, with various approaches and results. Again, by summarizing the related works we look for the best, yet simple, option to get the chord sequence for our complexity analysis.

### 3.2.1 Fujishima and pattern matching

Takuya Fujishima [4] was the first one to design chord transcription algorithm, and has also introduced the common technique of using DFT to obtain **pitch class profiles** (chromas). He has used simple summing of the related frequencies to obtain the chromas. Then Fujishima chooses 27 commonly used musical chords for each root pitch (amongst them major, minor, augmented or diminished chords and chords with common added dissonances) – we refer to this set as the **chord dictionary** – and matches each chroma sample to a chord in the dictionary. The **scoring** algorithm that Fujishima proposes uses Euclidean distance between the dictionary chord and the chroma – he calls it the **nearest neighbor method** – the nearest chord (with the best score) is selected. Note that, there are many different ways to match the chroma sample to the dictionary other than Euclidean distance,

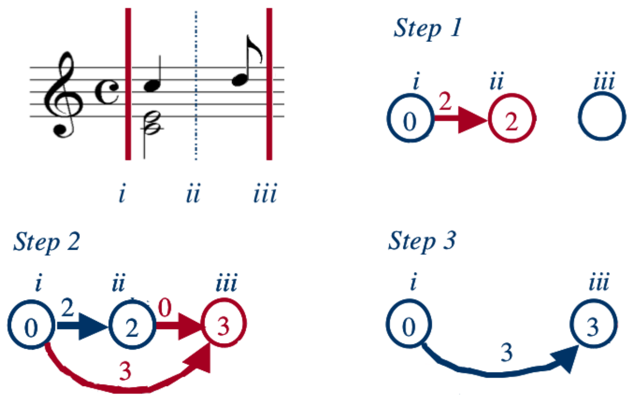



Figure 9: Segmentation as proposed by Pardo and Birmingham [16] – if the score of the chord is increasing or stays the same by adding a note, the note is added to the chord without segmenting

and we summarize them in one of the following sections. Fujishima also proposes simple **smoothing** to merge adjacent chromas as the heuristic to improve overall performance.

### 3.2.2 Chordal analysis

Bryan Pardo and William P. Birmingham [16] have proposed an algorithm that aims to find precise chord boundaries between the chords. When the new tone or multiple tones are played in the musical piece, decision has to be made, whether the tones remain as the part of the previous harmony, or whether the harmony changes at that point. The **segmentation** algorithm by Pardo and Birmingham considers both cases – the previous harmony together with the new tones is matched to the chord dictionary, as well as the situation where two separate harmonies are formed. Then the algorithm greedily selects the best option through analyzing a directed acyclic graph (DAG), thus leaving the locally correct segmentation behind, see figure 9. Using MIDI as the input simplifies the detection of the start time of the notes.

### 3.2.3 Music harmony analysis improving chord transcription

De Haas, Magalhães and Wiering [3] have described, how music harmony analysis can improve chord transcription algorithms. They focus on the point, where pattern matching shows, that multiple candidates from the chord dictionary have similar scores. They proceed to compare two systems – one that simply chooses highest scoring candidate, and the second one, that lets the tonal harmony rules decide, which candidate is the best. The authors have found statistically significant improvement, when the tonal harmony analysis was used. Later in the discussion they compare different approaches from MIREX 2011 challenge results. The algorithms proposed only have around 75% accuracy in finding the correct chords compared to ground truth. The only algorithms returning accuracy more than 74% were HMM-based machine learning approaches and the algorithm from Bas de Haas et al. However, as we have discussed in the introduction, HMM-based algorithm is likely to behave accurate on the genre it has been trained on and less accurate on the other genres, whereas harmony-based algorithm is likely to behave the same way in different genres.

Work from De Haas et al. is also amongst the few that actually shows a way to describe harmonic complexity, even though it was not the aim of the work. The presented Haskell-based system HarmTrace[11] uses tonal harmony to select the best chord candidate, by deriving a tree structure explaining the tonal function of the chords in the piece, see figure 10. It tries to label the chords in accordance with the basic **T – S – D – T** harmonic progression, enforcing that the piece needs to be organized as a sequence of tonics and dominants, optionally preceded by subdominant. Instead of main functions, a parallel may be used. If it is not possible to derive such tree, and a node needs to be deleted or inserted in order to achieve a valid progression, HarmTrace calculates the number of errors and chooses the chord candidate based on the lowest local number of errors in harmonies. Such model, if used globally, can be used to derive a basic harmonic complexity of a

[11] http://hackage.haskell.org/package/HarmTrace-2.0

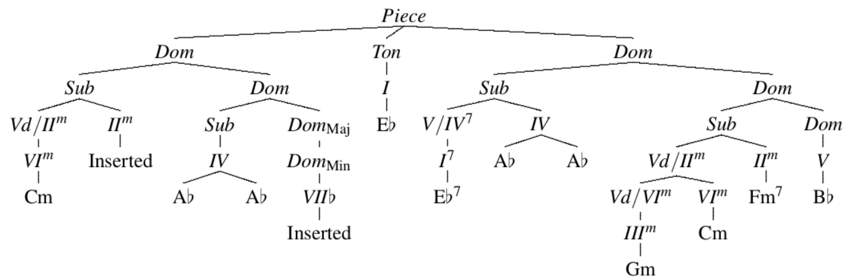



Figure 10: Harmony analysis as proposed by De Haas et al. [3] – the HarmTrace system deriving a tree describing the tonal functions of the chords, excerpt of the analysis of *The Long And Winding Road* by The Beatles

piece, e.g. by outputting the total number of errors (more errors – higher complexity).

Another thing we might learn from is the straightforwardness in using the groundwork techniques (usage of Vamp plugins and Euclidean distance) so they can focus on the main objective – proving that harmony improves chord transcription.

#### 3.2.4 Working with added dissonances and tone clusters

All chord transcription algorithms described above work with a smaller subsets of chords commonly used in music. That is to no surprise – most of the music is built on such subsets. Moreover, this approach can deal with the melody tones that are not part of harmony – non-chord melodic tones simply would not be matched because the chord dictionary does not contain such chords with added dissonances.

However, if we are to evaluate the true harmonic complexity, we should be interested in more complex dictionary. Chords with added dissonances are commonly used in the modern compositions, moreover, we might also benefit from letting the tones of the melody into our analysis. According to Zika [25], melody may be in harmony with its accompaniment, or it may create dissonances and additional tension towards the next movements. It would be interesting if our complexity analysis would differentiate two songs with the same music accompaniment, but one having more dissonant melody. Having broad dictionary with a lot of dissonant chords is therefore desirable.

In our previous work, Ear training application [13], we have created a system *Chordanal*, that was able to name all harmonies from chords to clusters. The aim was to create an interactive application for music conservatories for the Ear training course. First, the student selects the lesson. Then he gets the chord assignment – the program plays the chord. Student's task is to write in the text field exactly what he or she hears. The student may use standard name for the chord, if possible. But usually, if the assignment becomes harder, the training works step-by-step and the student only writes what he is sure to hear, e.g. the boundary interval of the chords. This way, he or she learns fast to recognize the musical sounds. Moreover, in our application we use the fact, that breaking down the structure, also the more complex harmonies may be named – chords with added dissonances may be denoted as the original chord plus all the intervals that create the dissonance. Chordanal standardizes such naming and given the chord with an added dissonance it can distinguish the chord and the dissonance, in multiple ways if possible.

First of all, parts of Chordanal system (since it is a Java object-oriented framework) help us work with the chords encapsulating them in the class, and then analyze what are the possible diatonic functions of the chords. Secondly, Chordanal system also helps us name all harmonies during the analysis, to provide more verbose output for the user. Re-using and broadening the system seems as a good option for our work.

To conclude this section, the best approach seems to be using pattern matching to a chord dictionary, like in the works presented. If possible, the results of harmony analysis should be used to determine the final chord sequence. And since we actually are interested in finding more dissonant chords, rather than choosing a common subset of chords, we broaden the dictionary as much as possible with the help of Chordanal.

## 3.3 Towards models for harmonic complexity

In this section we discuss, what are the options to evaluate harmonic complexity once we have the chord sequence. The HarmTrace system developed by De Haas et al. [3] shows a simple way how to evaluate complexity of the musical piece. There are other models, that relate to harmonic complexity, since creating various models is the core study of not only MIR, but the modern music theory itself. Lots of works have been done on **tonal tension** (see Lerdahl and Krumhansl [10]). However, tonal tension falls more under music perception – and we want to obtain theoretical model. Another reason why we might not be able to reuse works of tonal tension is, that it focuses on the distance from tonic, whereas we might consider the tonic, subdominant and dominant as equivalent, meaning that they are all three the fundamentals of any simple musical piece. Nevertheless, the works on tonal tension can point us in the right direction.

Other types of models that closely relate to music complexity, are models for **chord distance**. Many musical models have already been proposed to describe the relationships between tones, chords or keys. We have already talked about using Euclidean distance to find the best match amongst the chord candidates. Much more approaches can be used.

The work by Rocher, Hanna, Robine and Desainte-Catherine from University of Bordeaux [20] summarizes 8 different chord distances and examines their performance when used for chord transcription in pattern matching. The work concludes, that particular type of distance may be good for particular applica-

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *(a) octave (root) level:* | 0 | | | | | | | | | | | | (0) |
| *(b) fifths level:* | 0 | | | | | | | 7 | | | | | (0) |
| *(c) triadic level:* | 0 | | | | 4 | | | 7 | | | | | (0) |
| *(d) diatonic level:* | 0 | | 2 | | 4 | 5 | | 7 | | 9 | | 11 | (0) |
| *(e) chromatic level:* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | (0) |

Figure 11: Basic tonal pitch space as proposed by Lerdahl [9], set to *C major*

tions, therefore we need to choose the chord distance type based on what we want to achieve. From their summary, we choose those chord distances, that seem the most useful for harmonic complexity.

### 3.3.1 Chord distance in tonal pitch space

Lerdahl [9] introduces the term **tonal pitch space**, a model describing distances between pitches, chords and keys. The model starts with the basic space. The different levels of the basic space are shown in figure 11. Then transformations of the basic space measure the distance between chords. Lerdahl proposes the chord distance of two chords $C_x$, $C_y$ from the possibly different keys $K_x$, $K_y$ to be calculated as:

$$\delta(x,y) = i + j + k$$

where $i$ is the distance between the keys $K_x$, $K_y$ in the circle of fifths, i.e. the number of moves in the circle of fifths at level (d), $j$ is the distance between the chords $C_x$ and $C_y$ in the circle of fifths, i.e. the number of moves in the circle of fifths at levels (a-c), and $k$ is the number of non-common pitch classes in the space $x$ compared to the space $y$.

### 3.3.2 Tonnetz and Neo-Riemannian theory

Another important model is that of geometric harmonic grid called **Tonnetz** (*tone network* in German), proposed by Hugo Riemann [18]. The idea, first described by Leonhard Euler, is to represent tonal space as a two-dimensional pitch space grid, see figure 12a. The relationships represented by the edges originate in the just tuning and have been adapted to mirror the fundamental rules of tonal harmony. These ideas are extended in the **Neo-Riemannian theory**. First proposed by David Lewin [11] [12], the triads may be modified using three basic transformations, see figure 12b. The R transformation exchanges a triad for its Relative, e.g. *C major* to *a minor*, the L transformation exchanges a triad for its Leading-tone exchange, e.g. *C major* to *e minor*, and the P transformation exchanges a triad for its Parallel, e.g. *C major* to *c minor*. Note the ambiguity in the *parallel* term – here, the parallel comes from the notation commonly used in USA, and means modifying *C major* to *c minor*, whereas the parallels how we defined them, based on original Riemann's German notation, yields modifying *C major* to *a minor*, which would be called *relative* in USA (the same ambiguity is in describing the keys). The triads are shown on Tonnetz as triangles (more complex chords and harmonic progression may be then visualized e.g. as proposed by Bergstrom et al. [1], see figure 12c) and Neo-Riemannian transformations are shown as inversions of the triangles around one of its edges. For more information, the reader may refer to Cohn [2].

We may conclude this section by stating, that there are plenty of models related to harmonic complexity, but the way how they can help evaluate the complexity of a musical piece was not yet described[12].

[12]There actually is an article defining music *space* complexity the same way as the computational complexity theory: *The Complexity of Songs* by Donald E. Knuth [7]. Knuth describes the space complexity of songs as linear, but finds interesting results for *Old McDonald had a farm* song, and even logarithmic and constant complexity for some modern popular songs. Although published as an inside joke on computational complexity theory, we can take the advice of using computational complexity as a measure for harmonic complexity. Even more importantly, we can quote Knuth on that the repetitions and refrains – or simply the *space complexity* – should not be forgotten when defining the harmonic complexity as well.

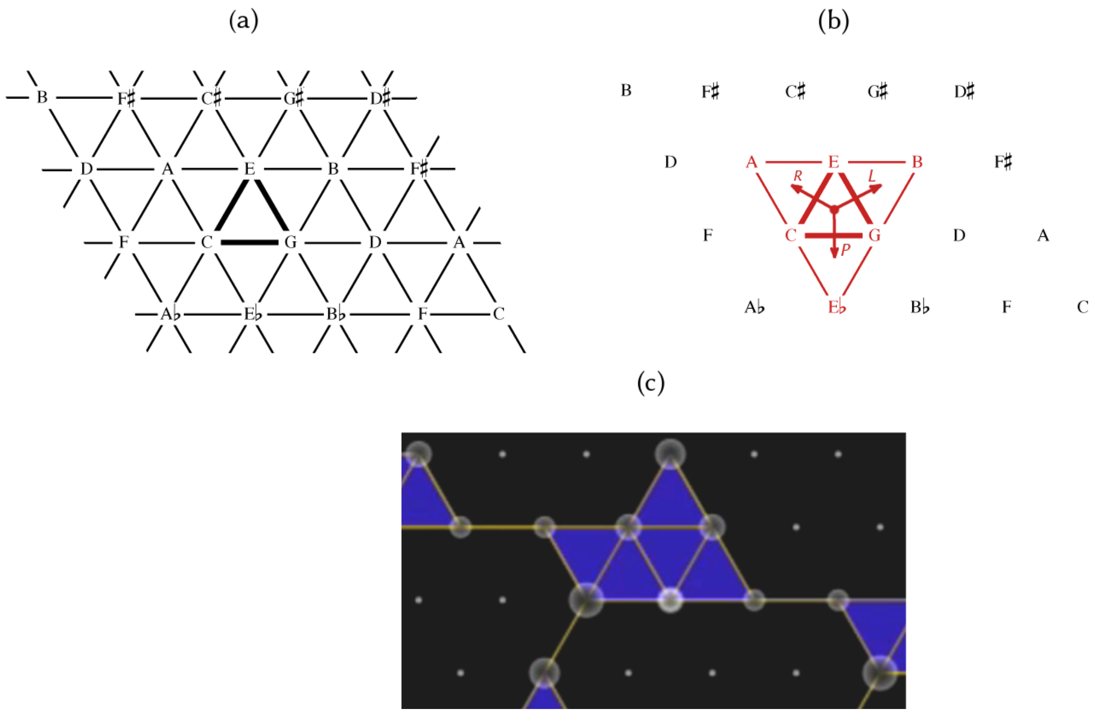



Figure 12: Using Tonnetz grid to visualize chords and describe distances in tonal harmony: (a) Tonnetz grid as proposed by Riemann; (b) Basic Neo-Riemannian transformations as proposed by Lewin [11] [12]; (c) Isochord visualization as proposed by Bergstrom et al. [1]

# 4 Choosing the techniques

In this chapter we summarize the chosen components for our system, using the knowledge from the previous chapter. The aim is to have the groundwork ready so that we may then simply plug in the model from chapter 5, the main part of this work.

From previous chapter we can conclude, that for groundwork, we prefer simple techniques rather than complex, because our main focus is to put new complexity model into practice rather than optimizing the little pieces for best precision or performance. The exception is choosing the chord dictionary. Rather then choosing a subset of commonly used chords, as seen in Fujishima [4] or De Haas et al. [3], we prefer to use broad dictionary to include also the dissonance of melody with the musical accompaniment into our analysis. This can be achieved by *Chordanal* system for parsing the complex harmonies presented in [13].

## 4.1 Harmanal system outline

We describe two variants of our system called Harmanal, based on the input method (WAV or MIDI).

### 4.1.1 WAV input

The outline for our *Harmanal* system to evaluate harmonic complexity of musical pieces is as follows:

1. We take WAV sound files as the input.

2. Feature extraction: We use Vamp plugins to extract the audio features – chromas and beats (NNLS Chroma Plugin, Bar and Beat Tracker Plugin)[13].

[13]For the simplicity, we do not yet work with the chord inversions, therefore we neglect the bass tone. However, since NNLS Chroma Plugin can extract also bass vectors, using of chord inversions is later possible in order to bring our complexity model to the next level.

3. Smoothing 1: We merge the chromas according to the beats, thus obtaining beat-synchronized chromas. The merging is done by averaging the chroma vectors between the two beats.

4. Chord approximation: We do pattern matching using the Euclidean distance to estimate the chord using the nearest neighbor technique – the best candidate is chosen. We choose the chord dictionary to contain all possible chords, chords with added dissonances and clusters made op of $n$ tones (maximal *size of harmonies*, formally defined in section 5.2). Specifying concrete $n$ depends on the implementation, but we may assume $n <= 12$, since working with pitch classes. We rely on *Chordanal* system to identify the chords with the increasing $n$. Having the maximum of tones $n$, the pattern matching simply means choosing the $n$ strongest chroma features. Since the features are floats mapped to $< 0, 1 >$, we take the $n$ highest floats and set them to 1, and set the other pitch classes to 0 to obtain the chord (as in figure 7). However, a threshold of $T$ is introduced to distinguish the important sounding tones from the ones that do not play significant role and are almost not noticeable in the harmony. So even though we are interested in additional dissonances, we set 0 for all features lower than $T$.

5. Smoothing 2: If adjacent chords are the same, we merge them into one. In the result, we obtain the chord sequence $\{C_i\}_{i \leq l} = c_1, c_2, \ldots, c_l$.

6. Complexity evaluation: Chordanal helps us to analyze the chord or cluster, extracting as many tonal-related informations as possible (root, possible keys, dissonances, etc.). We then use a complexity model on $\{C_i\}$ to determine the complexity of the piece.

7. The other output of the Harmanal system is, with the help of Chordanal, labeling the chords to provide verbose output for the user. As we describe in [13], there are multiple ways to perceive and label a single chord. However, during the analysis, based on the complexity model it uses, Harmanal

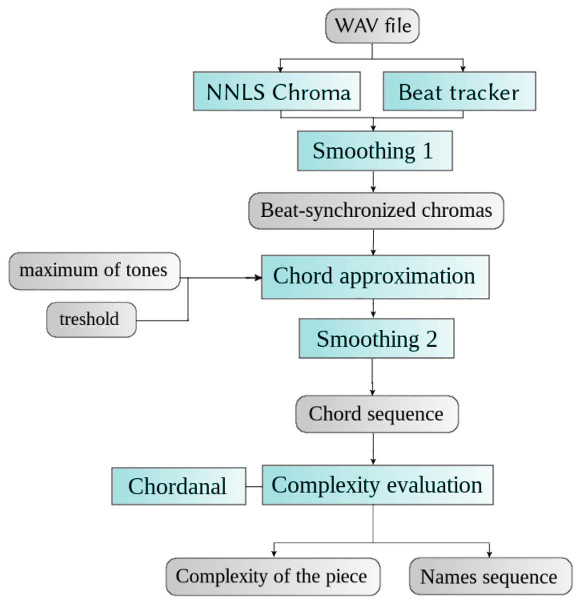



Figure 13: Harmanal: System for evaluating the harmonic complexity of musical pieces

chooses one possibility for the chord label that fits the best for the analysis, or two possibilities, if the chord is having a role of *pivot chord* in the diatonic modulation. Therefore, outputting the sequence of chord names $\{NAMES_i\}$ is a by-product of the analysis.

The schema of the Harmanal system is depicted in figure 13.

### 4.1.2 MIDI input

Thanks to the flexibility of the object oriented framework we work with, and due to the desired flexibility of our application, there is also another variant of Harmanal system that parses a real-time MIDI input from MIDI instruments (figure 14):

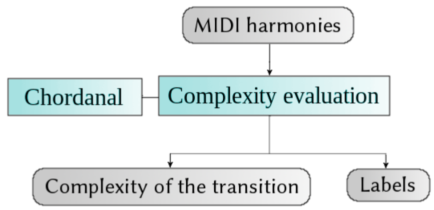



Figure 14: Harmanal, variant 2: System for evaluating the harmonic complexity of two MIDI harmonies

1. We take MIDI signal as the input to obtain two separate harmonies, $c_1$ and $c_2$.

2. Complexity evaluation: Chordanal helps us analyze the two harmonies. Here, we do not define the maximum of tones, because, fundamentally, Chordanal and the complexity model works for as many as 12 pitch classes sounding together (in the first variant the maximum of tones was used for optimizing the performance). We do not use the threshold as well because we suppose that if the musician played the tone, he or she wants to have it involved in the analysis. Our complexity model analyzes the complexity of the transition from $c_1$ to $c_2$.

3. Harmanal system also outputs, with the help of Chordanal, the preferred and all possible labels for $c_1$ and $c_2$.

## 4.2 Choosing the complexity model

The only yet undefined step is what complexity model we should use (step 6 in the WAV variant or step 2 in the MIDI variant of Harmanal system). There's a possibility to use already known evaluations, such as Lerdahl's chord distance, because it seemed most related to what we want to achieve. Applying it to adjacent chords in the sequence $\{C_i\}_{i \leq l}$ and then aggregating the distances would output the complexity of the piece.

However, we do not want to neglect some of the things specific for tonal harmonic complexity (such as usage of tonic, subdominant and dominant, usage of the parallels, added dissonances, etc.) so we prefer building a new tonal harmony model specific for complexity, but still based on Lerdahl's tonal pitch space. In the next chapter we propose and describe this new model in its basic form.

Also, there seem to be other important aspects of harmonic complexity, than just the tonal harmony view (for example, how much the harmony patterns are repeated) Note that, the complexity model in Harmanal is interchangeable for any other model, resulting in flexible environment for complexity analysis. We show the other possible complexity models in the chapter 5.

# 5 The TSD distance model

In this section we focus on one concrete aspect of harmonic complexity – the complexity of the harmony transitions based on the tonal harmony functions. Having two harmonies, our aim here is to find out what function the harmonies represent, and how complex is the transition between them. Therefore, we name the new model the **TSD distance model**, as an acronym for tonic, subdominant, dominant distance model. Note that, it is *not* an exhaustive tonal harmony model – it does not include modulations, nor voice leading, which are both fundamental parts of the tonal harmony. Modulations and voice leading should be treated in separate models, because they work on different levels of the tonal pitch space. The TSD distance model can nevertheless be used to analyze the complexity of the pieces modulating to different keys. For more information on this separation of complexities and the summary of all other harmonic complexities, see chapter 5.

## 5.1 Basic idea

The basic idea is to measure, how far the harmony transitions of the piece deviates from the basic **T – S – D – T** progression. Slightly differently from the tonal tension [10] – we are not interested in how much the music deviates from the tonic; we rather study how much the music deviate from the transitions between all three functions. To describe it even more, the main difference is not focusing on *where we are* in the progression, but how this is different from a simple progression. For example, if we are in basic tonic and move to basic subdominant, the tonal tension rises. If we then move to basic dominant, the tonal tension rises again. But our complexity should remain the same.

We build this model on the overtone series rules and the consequent harmony rules by Riemann [19]. The transitions from the basic harmonic progression **T – S**, **T – D**, **S – T**, **D – T** are indeed according to Riemann the *simplest harmony* *transitions*, due to the simplicity of the fifth interval. If dominant is built on the fifth degree *above* tonic, then subdominant is built on the fifth degree *below* tonic, and from that we derive not only the name for subdominant, but also some „equality of rights“ between subdominant and dominant, when it comes to the *usage* in music, as opposed to the tension. In conclusion, if only these transitions are used, we should consider the harmony very simple. The special case when the music stays at one function all the time is, again, out of the scope of this model and is briefly referenced in chapter 5.

We should take a closer look to the transitions **S – D** and **D – S**.

- The former, **S – D** is described by music theory as more difficult than the fifth interval transition, because the interval between the root tones is a whole tone. However, since we have already described that the subdominant has the same „right“ to be in the piece as the dominant, we can denote this transition as being very common in music as well, and it indeed is, as the part of the *basic harmonic progression*.
- The latter, **D – S**, should not be used based on the rules of original tonal harmony. However, as we have stated in the 2nd chapter, theorists allow different exceptions to this rule, and nowadays we might listen to this transition quite a lot.

The point is, that we *might* want to differentiate these transitions from the fifth interval transitions, however, we rather *do not* want to, because they can still be considered as the part of the basic T – S – D – T progression, and we want to measure the distance from that progression.

What we therefore focus on are the parallels, or **modifications** of these main functions, or the harmonies that can be created by adding dissonances to them. Graphically, we may imagine the basic idea like in the figure 15. If the music stays in the T – S – D triangle, we consider it simple. However, if it deviates from

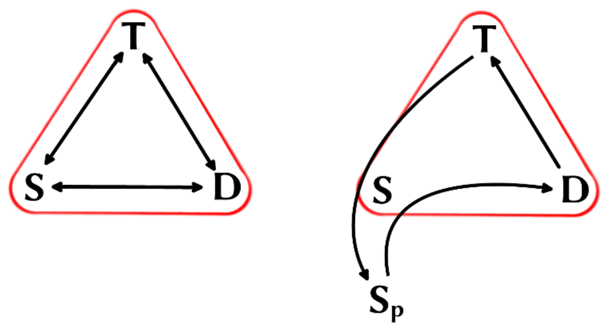



Figure 15: Basic idea of the TSD distance model: on the left the simple T – S – D progressions; on the right more complex progression using modifications

the triangle (in the figure using the parallel instead of *S*), we assign it a higher complexity. In the next sections we show how to evaluate such deviations.

*There is another reason why we did not assign any complexity to the transitions between T, S and D functions, that we may now fully understand: These transitions are principally different from the process of modifying a function. If we would, perhaps later, want to assign some complexity amongst them as well, we should treat it differently from the rest of the model.*

## 5.2 Formal definition

Before we go into details, we formally define the TSD distance model and other terms that we work with. For simplicity, we may use some well-known terms without definitions – we refer to Hoprcroft, Ullman and Motwani [5] for more information.

Also, we intuitively use the previously defined terms from music theory, that the reader can find in chapter 2. More formally, we denote the following terms and labels:

- *chromatic scale* $A$ as a set of 12 pitch classes: $A = cc\sharp dd\sharp eff\sharp gg\sharp aa\sharp b$.
- *tone* as the member of a chromatic scale, $t \in A$. The tones therefore obtain **pitch class** (relative) values.
- *harmony* as a set of *tones* and a subset of a chromatic scale, $h \subseteq A$. We commonly label it with letter $h$ and denote: $h = t_1 t_2 \ldots t_n$ where $t_1 \ldots t_n$ are the tones in the harmony. The *size of the harmony* is denoted $|h|$ and represents the number of its tones, $|h| = n$.
- *harmony universe* $U$ as the set of all possible harmonies: $U = 2^A$. (Note: Using the same terminology we can refer to $A$ as the *tone universe*.)
- In our work, we use the terms *key* and *scale* interchangeably, both referring to a subset of $A$. For simplicity we specify that both keys and scales contain exactly 7 tones (so they can be treated as 7-tones harmonies). We also use the key and scale names from the chapter 2, so that *C major* is a valid label of the key or scale, and we can use notation such as the tone $t \in C\ major$. If we do not mean a specific key, we commonly label the keys using $k$ letter and scales using $s$ letter.
- *key universe* $K$ as a set of all possible diatonic keys, 12 major and 12 minor.
- *harmonic function universe* $F$ as a set of possible basic harmonic function labels: $F = \{T, S, D\}$ for tonic, subdominant and dominant accordingly.
- *harmonic function* as the element of $F$, $f \in F$. (Note: Harmonic function is therefore not a harmony itself, only a label that we can use along with harmonies, e.g. harmony $h$ has a function $f$. However, given the key $k$ and the function $f$, the tonal harmony specifies a harmony made out of 3 tones that represents this function in the given key. Therefore we may, usually in the text, speaking of harmonic functions in a key, refer to a specific harmony. See chapter 2 for details.)

Although when displaying the content of a *harmony*, $h = ceg$ we can use different order of tones, still meaning the same harmony, in our work we strictly work with the notation where the tones are displayed in order of the chromatic scale. We may also use a *pitch class profile* format, 12-dimensional binary vector: $< 1,0,0,1,0,0,0,1,0,0,0,0 >$, to specify the harmony.

**Definition 1.** ***TSD distance model*** *is a tuple* $(T, P, roots, c, tc)$*, where* $T$ *is a finite alphabet,* $T \subseteq A$*;* $P$ *is a finite set of rules,* $P \subseteq_{finite} A^* \times A^*$*, for every rule* $u \to v \in P : |u| \leq |v|$*;* $roots$ *is a finite function:* $U \to 2^{K \times F \times U}$*;* $c$ *is a finite function:* $U \to \mathbb{N}$ *and* $tc$ *is a finite function:* $U \times U \to \mathbb{N}$*.*

The definition may induce many questions, but we will go step by step to answer them all. First of all, the reader might have noticed some similarity between the TSD distance model and context-sensitive grammars. We indeed designed it to behave similarly to the CSG, with the difference, that there is no nonterminal set, no start symbol, and we have functions *roots*, *c* and *tc* providing us an extra functionality. The reader might benefit from following section helping understanding the main idea of the model.

### 5.2.1 Understanding the formal model

The high-level idea of the model is, that instead of the start symbol, we use the *start sentential form*, which is represented by the basic harmony function, i.e. $r = ceg$ for the basic tonic function of *C major* key. We will call these basic harmonies, that can form the start sentential form, the **root harmonies** for a given key and denote them with the letter $r$. Then, using the rules from $P$, we can perform a derivation of the given harmony $m$ (modification of the function): $r \to h_1 \to h_2 \to \cdots \to m$. We need to keep in mind, that these derivations are key-specific and look differently in different keys. Such derivation can be seen on Example 1 on page 60 and seen graphically on Figure 16 on page 63, and will be described more in details.

The aim therefore is to derive the modification from the basic harmony function. Obviously, both of these harmonies should be labelled with the same harmonic function label ($f$) and we should have some algorithmic way to find this label and to determine the basic function to start with. In other words, for a given harmony $h$ we need an algorithm that will return a root harmony $r$ that the derivation can start with. For this exact purpose, we describe the function from the definition of TSD model, $roots$. But, to provide us all the neccessary information, the function $roots$ outputs for a given harmony not only the root harmony, but also the key and the function label, because we know that the process is key-specific and the function label can be useful, too. The function $roots$ is formally defined in section 5.2.2.

Notable difference is, that each derivation depends on the key $k$. Concretely, we need to have different sets of rules for each key. As the reader will find out, these differences are very straightforward, caused only by the different sets of tones in the key. However, for clarity and for underlining the difference between the TSD distance model and formal grammars, we will always refer to the derivation of the harmony $h$ with parameters $k$ (key) and $r$ (start sentential form). The derivation is formally defined in section 5.2.4.

As for the other two functions, $c$ is called *harmony complexity* and it returns the static complexity of the given harmony as the length of the derivation of the harmony. We formally describe harmony complexity in section 5.2.3. The other function, $tc$, is *transition complexity*. It takes two harmonies and outputs the complexity of the transition between them. It is little bit more difficult to describe, but we will have enough knowledge to formally describe it in section 5.2.5.

In the following, we will work with the concrete model: $TSD_{BASIC}$. Unlike with the family of TSD distance models, we will precisely describe how all the functions of this model work. We use the *basic* label, because we are aware that more functionality can be added later. In certain moments, we make some proposals for the model $TSD_{COMPLETE}$. One of the proposal we can make from the beginning is, knowing that $TSD_{BASIC}$ does not take the bass tone into the account and therefore it can not work with chord inversions. A more complete model $TSD_{COMPLETE}$ could take all the rules from $TSD_{BASIC}$ and add the functionality for inversions. and for the future works. However, a specification of such $TSD_{COMPLETE}$ model is out of the scope of this work and the more simple model $TSD_{BASIC}$ is sufficient to achieve our goals.

**Definition 2.** *$TSD_{BASIC}$ is a TSD distance model:*

$$(A_{BASIC}, P_{BASIC}, roots_{BASIC}, c_{BASIC}, tc_{BASIC})$$

*Where $A_{BASIC} = A$, and the set of rules $P_{BASIC}$ and the functions $roots_{BASIC}, c_{BASIC}, tc_{BASIC}$ are defined in Definition 10, Definition 3, Definition 6 and Definition 16.*

Now we will, as collecting puzzle pieces, step by step describe all the remaining elements of $TSD_{BASIC}$.

### 5.2.2 Finding the root harmonies

First we treat the problem of how to determine, what is the root harmony of the harmony $h$ (that goes in hand with determining the function label $f$ for $h$). The algorithm is simple: Analyzing the tone material of $h$, and comparing it to $T$, $S$, $D$ harmonies of different keys (let us denote the basic harmony function with label $f$ in the key $k$ as $k(f)$; we need to compare $h$ with $k(f)$ for all combinations of $k$ and $f$). We may conclude that, if $h$ contains all three tones of $k(f)$, the harmony $k(f)$ is its root harmony.

From Zika [25] and other harmony textbooks we can see evidences, that somesometimes, even two chord tones are sufficient to represent the basic harmony function. Zika specifies that in some cases, the tonic triad in the end of the musical phrase contains only **root tone** and the **third**. Note that, it can still retain the major or minor character. We also find evidences, that in particular cases, also **root tone** and the **fifth** may be considered an incomplete triad, although in this case we can not state the character. Moreover, the stand-alone I., IV. or V. degree can intuitively represent the function too, if no other tones are present. So if the function triad is not present in the harmony *h*, we may lower our boundary and look for only portions of the function triad. On the other hand, if we have found the match for multiple tones from the function, there is no reason to look for smaller matches. Since 1 tone is too small of an evidence for the presence of a function, we consider one-tone root harmony match only if there are no multitone matches, in all the other keys.

We find the confidence in such approach in Leoš Janáček's conception of *chords with added dissonances*, theoretically described by Volek [23]. Janáček promotes, that by adding certain amount of dissonances, the chord still retains the same function, provided that these dissonances can not be described otherwise, diatonically. The good measure is, that the number of added dissonances should not outrank the number of tones in the root harmony. However, there are cases (large clusters) where no such configuration can be found – in that case we choose the largest possible root harmony again. So we use the measure from the previous paragraph – 2 tones (interval) root harmony is sufficient, if no 3-tones root harmony is found, and single-tone root harmony is considered only if there is no 2 or 3-tones root harmony[14].

We thus can easily design the $roots_{BASIC}$ function (as a part of $TSD_{BASIC}$ model) programatically, see the simple pseudocode. To satisfy the added dis-

[14]We must mention here, that in this section, even though there have been some related studies, we are considering rules based more on experience of the composers rather than on fundamental rules of tonal harmony or acoustics. That can be dangerous, given that the composers' and theorists' views may differ in minor points and our model may loose its generality, as we have discussed in the section 2.1. One of the proposals for $TSD_{COMPLETE}$ can therefore be to minimize the effect of rules that might be considered as not established.

sonances rule we stop searching for root function in a certain key once the dissonances outnumber the root function tones. That decreases the number of results, positively influencing the outcome, because we do not want big harmonies to be classified as having a root harmony only one tone.

```
array roots(harmony h)
begin
    results = empty

    // adding 3 or 2-tone root harmonies
    foreach k in K
        foreach f in F
            if (h contains all tones of k(f))
                add <k, f, common tones> to results
            else
                if (h contains root+third or root+fifth of k(f))
                     add <k, f, common tones> to result

    if (results found)
        return results

    // adding 1 tone root harmonies
    foreach k in K
        foreach f in F
            if (h contains root tone of k(f))
                add <k, f, common tone> to results

    return results
end
```

Formally, we may describe $roots_{BASIC}$ as a set of homomorphisms $h_{k,f}$, one for each combination of the key and the function label, that leaves the tones of the $k(f)$ in the harmony, and erases the remaining tones. In the following definition we use homomorphism for tones generalized for harmonies the same way that it is used for words in [5]. For simplicity, we do not formally describe the optimization for stopping the search. Leaving all the results in the formal model is fine, since we do not look for optimizations there, as opposed to the implementation where we would be better off looking for more optimal solutions.

**Definition 3.** *Function $roots_{BASIC}$ is the function $U \to 2^{K \times F \times U}$ with the definition:*

$$roots_{BASIC}(h) = \bigcup_{k \in K} \bigcup_{f \in F} (k, f, h_{k,f}(h))$$

*Where $h_{k,f}$ is the homomorphism that $\forall t \in A$:*

$$h_{k,f}(t) = t \Leftrightarrow t \in k(f)$$

$$h_{k,f}(t) = \varepsilon \Leftrightarrow t \notin k(f)$$

Sometimes it will be handy to access only the keys, functions, or harmonies from the *roots* output – we use *projection* with the label of the desired value for clarity:

- $\pi_K(roots_{BASIC}(h))$ selects the first values from the *roots* output (keys)
- $\pi_F(roots_{BASIC}(h))$ selects the second values from the *roots* output (functions)
- $\pi_U(roots_{BASIC}(h))$ selects the third values from the *roots* output (harmonies)

Defining the root function will allow us to define the root harmonies formally. Knowing that the root harmonies form only the subset of $U$, we define:

**Definition 4.** ***Root harmony universe*** *$R$ is the set defined as follows:*

$$R = \{r \in U \mid \exists h \in U; \pi_U(roots_{BASIC}(h)) = r\}$$

**Definition 5.** *Given a harmony $h$, we call all harmonies $r$ such that $r \in \pi_U(roots_{BASIC}(h))$ the **root harmonies** of the harmony $h$.*

Due to the table character of the *roots* output, we might (and in the application we also do) approach it as a database. For our formal language we therefore also describe notation for *selection* in addition to projection, similar to relational algebra:

$$\sigma_{K=k_1,\ F=f_1}(roots_{BASIC}(h))$$

Some analysis of our root finding follows:

**Theorem 1.** *For each harmony in $U$ there exists a root harmony.*

*Proof.* In the Definition 3 of roots function we project the given harmony with the homomorphism for every combination of function label and key, that means also for $T$ in every key. Amongst these tonic harmonies, all tones of the chromatic scale $A$ are present. The homomorphisms keep the tones of these tonic harmonies in the given harmony, and that means that at least one homomorphism will leave non-epsilon output. □

However, there are some harmonies that have only trivial root harmonies $r$ with $|r| = 1$, which does not „look good" for tonal harmony analysis, but the following theorem is stating that there are not many of such harmonies (and for those that have trivial root harmonies it is reasonable to have them).

**Theorem 2.** *The only harmonies $h$ having root harmonies $r$ with $|r| = 1$ are the ones with structures: $p1$; $m2$; $M2$; $tritone$ and $m2, M2$[15].*

*Proof.* Listed basic intervals $m2$; $M2$; $tritone$, are the only ones not present in the major or minor triad or its inversions. If we want to find more harmonies not present in a diatonic triad, we can combine them together – thus adding only $m2, M2$ harmony, because other combinations lead to an interval present in diatonic triad. Adding anything else to this set (not counting *unisone*) would lead to

[15] By saying harmony with a *structure* we do not say the exact pitches of the harmony, but we mean family of harmonies with tones in specified intervals. Comma-separated intervals such as $m2, M2$ denote a harmony built from all of these intervals from the lowest tone, so in this particular case e.g. a harmony $cc\sharp d$, $c\sharp dd\sharp$, etc. – it is common notation we have used in Chordanal system, see [13].

introduce an interval present in diatonic triad, therefore matching some interval in tonic, subdominant or dominant function. □

**Consequence 1.** *Nice consequence is, that all the remaining harmonies (= vast majority) have at least 3 non-trivial root harmonies (because for any third, fourth, fifth or sixth interval there is a key in which tonic matches, another key in which subdominant matches and the third key where dominant matches). The same – that always some tonic, subdominant and dominant matches – works also for trivial root harmonies, so we can make a stronger statement than the* ***Theorem 1*** *in a way, that each harmony has at least 3 root harmonies.*

We should be careful though, because if there are too many matches, we might encounter some time complexity issues. Note also how we were working with intervals in the proof – we work in $\mathbb{Z}_{12}$ group, so when considering an interval, we as well consider its inversion (e.g. major second, minor seventh).

### 5.2.3 Harmony complexity

First important complexity that we are able to evaluate is the complexity of a harmony. It represents the static complexity of a single harmony, which will later help us to define the complexity of a transition. For $TSD_{BASIC}$ model it is nothing else than the *distance* from basic harmony functions, i.e. the length of derivation from a root harmony to the harmony. Using the *BASIC* label leaves the door open for further definitions of harmony complexity (it is considerable also to use different static measures already defined in the literature, e.g. the *surface tension* from Lerdahl [10]).

**Definition 6.** *Harmony complexity in $TSD_{BASIC}$ model: $c_{BASIC}(h)$ is the minimal length of derivation of $h$*[16].

As we have mentioned earlier, by derivation we mean the same as derivation in CSG, with the difference that $TSD_{BASIC}$ implicitly takes the start sentential form

[16] Harmony complexity defined this way is actually the *computational time complexity* of the harmony in our model.

from $\pi_U(roots(h))$. Another important difference is, that to obtain the minimal derivation, we will need to compare derivations starting from all root harmonies in $\pi_U(roots(h))$, given the respective keys from $\pi_K(roots(h))$. The harmony complexity is then found as the minimal length of derivation amongst all of these derivations. We clarify this approach in the following section using examples and a definition.

Sometimes we might want to look for the complexity of the harmony *within* the specific key – the syntax then is $c_{BASIC}(k,h)$. That means that we only consider the derivations parametrized by the given key. If we want to be even more specific, we may specify the key and the function, thus obtaining the complexity of the harmony from the the specific function in the key: $c_{BASIC}(k,f,h)$.

### 5.2.4 Derivation explained

Let us consider the following example:

$$k = C\ major$$
$$r = ce$$
$$h = cef\sharp g\sharp$$

According to Janáček's added dissonances, adding a tone can be considered as a fundamental operation that we can do multiple times, provided that the chord can not be described otherwise. Therefore we propose the **ADD** rule, adding a tone to the harmony (formally we define it at the end of the section).

However, according to Lerdahl [9], if we are in the certain key $k$, there are different levels of pitch classes that we should take into consideration, from chromatic up to root level as in figure 11. We thus propose for $TSD_{BASIC}$, that at least the chromatic and diatonic level should be taken into consideration, and distinguish if the *ADD* rule added a diatonic or non-diatonic tone. In accordance with other established practice in tonal harmony, *alteration*, we may do such distinction: alteration is a chromatic process in diatonic system, that allows certain diatonic tones in major or minor scale, to be altered a semitone up or down. $TSD_{BASIC}$ can generalize otherwise quite specific rules of alteration and let all diatonic tones have a possibility of alteration, with the exception of root.

We thus propose an **ALTER** rule, moving the tone of the harmony a semitone up or down. There are several restrictions to $ALTER$ rule:

1. We can not alter the tone of the root harmony. We would then weaken the function of the harmony.

2. It's not possible to alter the diatonic tone of the scale resulting another diatonic tone, i.e. in *C major* it's not possible to alter tone *e* or *b* up.

Thus, we let the *ADD* rule operate only on diatonic tones, whereas $ALTER$ rule would be the only chromatic process in the derivation. Now we can also see, why the derivation depends on the key $k$. To show the resulting derivation, let's get back to the example from the beginning:

**Example 1.** *The derivation of $cef\sharp g\sharp$ harmony in C major starting from root harmony ce in $TSD_{BASIC}$:*

$$r = ce \xrightarrow[ADD]{} cef \xrightarrow[ALTER]{} cef\sharp \xrightarrow[ADD]{} cef\sharp g \xrightarrow[ALTER]{} cef\sharp g\sharp = h$$

*The complexity of the $cef\sharp g\sharp$ harmony can be then found as the number of steps in our derivation. Note that, there are multiple derivations even for the given key and root harmony, depending on what is the order of tones that we derive, for example:*

$$r = ce \xrightarrow[ADD]{} cef \xrightarrow[ADD]{} cefg \xrightarrow[ALTER]{} cef\sharp g \xrightarrow[ALTER]{} cef\sharp g\sharp = h$$

*is another derivation of the same harmony. Normally, we would need to check all the possibilities. In our example it is however evident that:*

$$c_{BASIC}(C\ major, T, cef\sharp g\sharp) = 4$$

*Since ce does not have matches for subdominant or dominant in C major, we also obtain:*

$$c_{BASIC}(C\ major, cef\sharp g\sharp) = 4$$

*However, here comes the „tricky bit“: even though it might not look like it, $roots(cef\sharp g\sharp)$ outputs as many as 9 different root harmonies, so we are able to make 9 independent derivations with different complexities for $cef\sharp g\sharp$, see table 5. Therefore, the resulting harmony complexity for $cef\sharp g\sharp$ is the minimum of the complexities:*

$$c_{BASIC}(cef\sharp g\sharp) = 3$$

| **key** | **function** | **root harmony** | **complexity** |
|---|---|---|---|
| *E major* | Tonic | $eg\sharp$ | 3 |
| *G major* | Subdominant | $ce$ | 3 |
| *B major* | Subdominant | $eg\sharp$ | 3 |
| *A major* | Dominant | $eg\sharp$ | 3 |
| *C♯ major* | Dominant | $cg\sharp$ | 3 |
| *C major* | Tonic | $ce$ | 4 |
| *G♯ major* | Tonic | $cg\sharp$ | 4 |
| *D♯ major* | Subdominant | $cg\sharp$ | 4 |
| *F major* | Dominant | $ce$ | 4 |

Table 5: Output of function $roots_{BASIC}(cef\sharp g\sharp)$ ($\rightarrow$ first three columns) and the according output of function $c_{BASIC}$ for the same harmony, given the key from the first and root harmony from the third column ($\rightarrow$ fourth column)

Now that we understand how the derivation works, we can benefit also from graphical representation of the harmony complexity, see figure 16 (complexity of modified subdominant, denoted *Sm*. The reader might remember the picture from the first section of this chapter (figure 15), where we have used a subdominant parallel. The model was designed in the way, that every *commonly used* modifications[17] such as parallels or counter parallels, or minor chord instead of major chord, etc., output the harmony complexity $c(h) = 1$ (the reader can easily verify). However, a little drawback of our $TSD_{BASIC}$ model might be, that they are not distinguished, and other chords with added dissonances also qualify for $c(h) = 1$. We thus leave a proposal for more advanced model $TSD_{COMPLETE}$ to implement the complexity with smaller granularity.

For clarity, we provide more formal definition of the whole derivation process. In the following we describe derivations similar to [5], but using harmonies instead of words for sentential forms. The reader can verify, that the use of harmonies for sentential forms is the same as using the words, because every harmony can be written as a word, with the tones in the order of the chromatic scale, as we have pointed out in the section 5.2. Another difference is, that the definition ot the rule in our model in fact represents a big family of rules, from which any rule can be used on sentential form.

**Definition 7.** ***Rule*** *in* $TSD_{BASIC}$ *is a binary relation* $\xrightarrow[NAME(k,r)]{}$ *on* $A^*$ *denoted by its NAME and parametrized by the key k and root harmony r.*

**Definition 8.** ***ADD*** *is a rule defined as follows:*

$h \xrightarrow[ADD(k,r)]{} g \Leftrightarrow (h = t_1, t_2, \ldots\ldots t_n) \wedge (g = t_1, \ldots t_i, t, t_{i+1}, \ldots t_n)$

*where* $(t \in k) \wedge (t \notin r)$.

**Definition 9.** ***ALTER*** *is a rule defined as follows:*

$h \xrightarrow[ALTER(k,r)]{} g \Leftrightarrow (h = t_1, \ldots\ldots t_i, t, t_{i+1}, \ldots t_n) \wedge (g = t_1, \ldots t_i, t_{alt}, t_{i+1}, \ldots t_n)$

*where* $(t \in k) \wedge (t_{alt} \notin k) \wedge (t_{alt}$ *is a semitone up or down from* $t) \wedge (t \notin r)$.

[17] The term **modification** of the function is used in some music theory literature as a common term for parallels and counter parallels of the function [25].

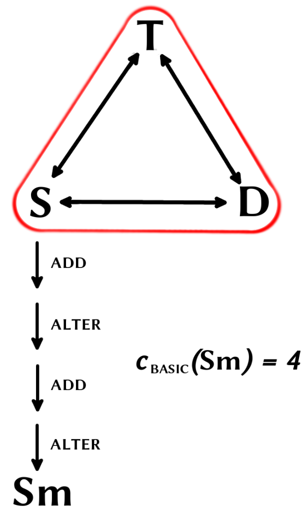



Figure 16: Graphical representation of harmony complexity

**Definition 10.**

$$P_{BASIC} = (\bigcup_{k \in K} \bigcup_{r \in R} ADD(k,r)) \cup (\bigcup_{k \in K} \bigcup_{r \in R} ALTER(k,r))\}$$

In the following, we simplify the writing of the rule – we write it without the brackets containing root harmony and key, but we will assume that every rule still *remembers* what is the key and what was the root harmony that the derivation started with. The important regulation of our model is, that when we obtain the root harmony $r$ from the $roots$ function, we also obtain the respective key $k$, and **only the rules with parameters k, r** can be used in the derivation. Other derivation would use different $r$ and $k$ based on the other output of $roots$ function, and so on, because in order to evaluate the harmonic complexity we need to take all derivations into consideration. We formalize these regulations in the next definition.

**Definition 11.** *In a TSD distance model* $(T, P, roots, c, tc)$*, we call* ***derivation*** *of a* *harmony h with the parameters k and r and denote $\Delta(k,r,h)$ such finite sequence of rules from P with parameters r and k, that starts with r and finishes with h, and $r \in \pi_U(roots(h))$ is the root harmony of h, and $k \in \pi_K(\sigma_{U=r}(roots(h)))$ is the respective key. Harmony r is called the* ***start sentential form*** *of the derivation. The* ***length of the derivation*** *is the number of rules used to derive h and is denoted as $|\Delta(k,r,h)|$.*

**Consequence 2.**

$$c_{BASIC}(h) = min(\bigcup_{r \in \pi_U(roots_{BASIC}(h));\ k \in \pi_K(\sigma_{U=r}(roots_{BASIC}(h)))} |\Delta(k,r,h)|)$$

So in conclusion a short quiz for clarification:

**Example 2.**
***Question:*** *What our model really does, while calculating $c_{BASIC}(h)$?*
***Correct answer:*** *It first calls the $roots_{BASIC}(h)$ function that enumerates the tuples in the form (key, function label, root harmony). It then performs all the derivations $\Delta(k,r,h)$ for every tuple from the table and outputs the smallest length of derivation it encountered.*

As we did also for the function *roots*, we provide a simple pseudocode for $c_{BASIC}$, that helps us analyze the computational time. The actual derivation is very straightforward, because we know what we start with and what the derivation should end up with, so we may deduct which tones need to be added (in pseudocode these tones are in the array **tones**). Some of these tones need to be altered after adding because they do not belong to the key. We do not go into details, but use following functions that do not increase overall time asymptotically:

- array getDifference(harmony h,harmony r) *// gets the array of tones present in h but not in r*
- boolean isDiatonic(tone t) *// returns* ***true*** *if the tone is diatonic in the key and* ***false*** *if not and it needs to be altered*

- tone getDiatonic(tone t) *// returns the diatonic version of the tone*
- harmony add(harmony s, tone t) *// adds the tone t to the sentential form s*
- harmony alter(harmony s, tone t) *// alters the tone from the sentential form s at the index i*

and it so the only question is what order of adding tones we choose. However, we can also notice (without proof) that with a given key and root harmony, the rules *ADD* and *ALTER* are designed that way, that each different derivation $\Delta(k,r,h)$ of $h$ yields the same complexity. For uniformity, we may choose to add tones in the order of the chromatic scale.

```
int c(harmony h)
begin
    int count = 0
    // initialize min to maximal possible value
    int min = 2|A|

    // table is an array of tuples K × F × U
    array table = roots(h)

    foreach k in π_K(table)
        foreach r in π_U(table)
        begin
            // tones is an array containing tones present in h but not in r
            array tones = getDifference(h,r)
            for i = 1 to (|h| - |r|)
            begin
                if (isDiatonic(tones[i])
                begin
                    r = add(r,tones[i])
                    count++
                end
                else
                begin
```

```
                r = add(r,getDiatonic(tones[i])
                count++
                r = alter(r,getDiatonic(tones[i]))
                count++
            end
        end
        if (count < min)
            min = count
    end

    return min
end
```

### 5.2.5 Transition complexity

Having described the complexity of a harmony we can move on to describe the complexity of a transition. What we are trying to achieve is, find out how „far" is the musical piece from the basic **T – S – D – T** progression. What we could simply do is sum all the harmony complexities in the piece – we would obtain a measure, how far are the functions from basic harmony functions. That is ok, but we would neglect seeing what happens in between the harmonies. It often happens, for example, that two harmonies share the tone material, so the transition is smooth for the listener, however both of the harmony complexities might be high because of the shared tones.

For that purpose, in our complexity model we have *transition complexity* function (TC). TC is closely related to chord distance (CD), however, in general, CD is not focusing on harmony functions. It would be nevertheless interesting to compare our TC with some CD algorithms from [20].

**Definition 12.** *Transition complexity* $tc_{model}(h_1, h_2)$ *evaluates the dynamic complexity between harmonies* $h_1$, $h_2$ *using the specified model.*

We first define a sample transition complexity, that we later modify to fit all of our needs.

**Definition 13.** *$tc_{COMMON}(h_1,h_2)$ is the sum of the lengths of minimal derivations of $h_1$ and $h_2$ from its nearest common ancestor in derivation. If nearest common ancestor can not be found for $h_1$, $h_2$, but there exists a common key $k$ in their roots tables, $tc_{COMMON}(h_1,h_2) = c_{BASIC}(k,h_1) + c_{BASIC}(k,h_2)$.*

**Definition 14.** *For two harmonies, $h_1$ and $h_2$, the **common ancestor in derivation** ($CA(h_1,h_2)$) is such sentential form, that appears in at least one derivation of both harmonies $h_1$, $h_2$.*

**Definition 15.** *For two harmonies, $h_1$ and $h_2$, the **nearest common ancestor in derivation** ($NCA(h_1,h_2)$) is such CA, for which the sum of the lengths of minimal derivations from CA to $h_1$ and from CA to $h_2$ is minimal.*

More easily stated, if we somehow (hypothetically) invert the rules *ADD* and *ALTER*, and we want to get from $h_1$ to $h_2$ as fast as possible, sometimes the path is not going from $h_1$ all the way to the root harmony, then potentially change the root harmony (which is for free) and then derive $h_2$. If they have a non-trivial CA (non-trivial = non root harmony), we can just invert the rules up til the CA and then derive $h_2$. If CA can not be found, but the harmonies $h_1$, $h_2$ share common key $k$, we encounter a transition between functions, so we go all the way to the root of $h_1$. Even though we really do not want to invert the rule to keep the model simple, and the *work* is therefore hypothetical, we may, again, see that this definition is pretty much the computational time complexity in our model. For a graphical representation of transition complexity, see figure 17.

**Example 3.**
*Derivation of $cef$ in* C major*: $ce \xrightarrow[ADD]{} cef$*
*Derivation of $cef\natural$ in* C major*: $ce \xrightarrow[ADD]{} cef \xrightarrow[ALTER]{} cef\natural$*
*$NCA(cef, cef\natural)$ in* C major *is $cef$ itself.*
*Length of minimal derivation of $cef$ starting from $cef$: 0*

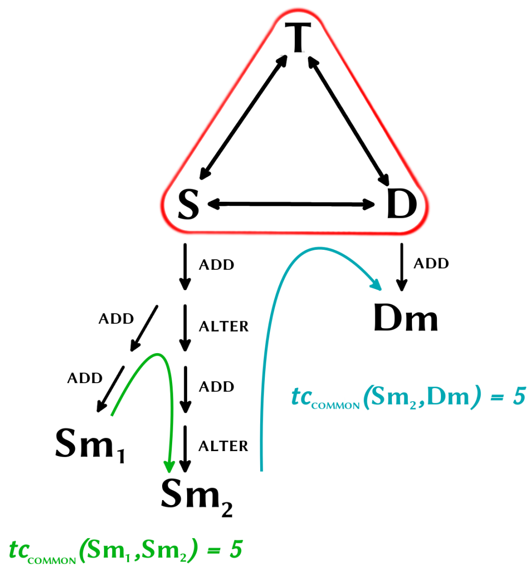



Figure 17: Graphical representation of transition complexity amongst the same function (in green) an in between two functions (in blue)

*Length of minimal derivation of* $cef\sharp$ *starting from* $cef$*: 1*
*Therefore:* $tc_{COMMON}(cef, cef\sharp) = 0 + 1 = 1$

**Theorem 3.** *If r is a root harmony for h, then* $c_{BASIC}(h) = tc_{COMMON}(r, h)$

*Proof.* Trivially, the NCA of $h$ and its root harmony $r$ is $r$. □

A pseudocode for $tc_{COMMON}$ is shown below. The algorithm starts with finding the common roots (tuples key, root harmony), and for these roots performs searching for a common ancestor. The biggest problem seems to be, that to search for a common ancestor properly, every possible derivation should be done for both harmonies $h_1$ and $h_2$ and the sentanial forms compared, which would lead to trying every order of added tones and the time complexity $O(|h_1|!|h_2|!)$. Luckily, there is an easy optimization: the tone material of $h_1$ and $h_2$ can be analyzed and the set of common *diatonic* tones that would need to be added can be found in $O(n^2)$.

The order of adding can be then done by the chromatic scale. We also alter the added tones, if they are altered in both harmonies. We show this finding of common rules as *findCommonRules(harmony h1, harmony h2)* function, returning an array of rules. In the pseudocode we then access the type of the rule number i (*rules[i]["type"]*, either *"ADD"* or *"ALTER"*, as well as the tone that needs to be added or altered (*rules[i]["tone"]*. Once we perform the common rules, we have found not only the CA for the given key and root harmony, but also the NCA for the given key and root harmony, because the *findCommonRules()* function finds all necessary tones and therefore the resulting sentential form is the largest possible. Then we only need to check all the different NCAs for the different tuples (key, root harmony) and choose the one with the highest complexity.

Amongst other functions that do not increase the time complexity we also use function *intersection(array table1, array table2) returning an array containing the intersection of two tables with tuples* $K \times F \times U$.

```
int tc(harmony h1,harmony h2) {
begin
    // common roots searching

    // table1 and table2 are arrays of tuples K × F × U
    table1 = roots(h1)
    table2 = roots(h2)

    table commonroots = intersection(table1,table2)

    // nearest common ancestor searching

    // we set the maxcomplexity to the minimum value possible
    int maxcomplexity = -1
    harmony nca

    int complexity = 0
    foreach k in π_K(commonroots)
```

```
        foreach r in πU(commonroots)
        begin
            array rules = findCommonRules()
            for i = 1 to |rules|
                if (rules[i]["type"] == "ADD")
                begin
                    r = add(r,rules[i]["tone"])
                    complexity++
                end
                else
                begin
                    r = add(r,getDiatonic(rules[i]["tone"]))
                    complexity++
                    r = alter(r,getDiatonic(rules[i]["tone"]))
                    complexity++
                end
                if (complexity > maxcomplexity)
                    nca = r
        end

    if (nca != null)
        // if common ancestor found, we only sum
        // the harmony complexities up to the ancester
        int complexity1 = c(h1) - c(nca)
        int complexity2 = c(h2) - c(nca)
        return complexity1 + complexity2
    else if πK(commonroots) != null)
        // no common ancestor but common key found,
        // so we sum the harmony complexities up to the root
        int complexity1 = c(h1)
        int complexity2 = c(h2)
        return complexity1 + complexity2
end
```

While for *harmony complexity* the **Theorems 1 and 2** with **Consequence 1** were the most important, because they show that we have relevant results for every harmony, in *transition complexity* the situation is different.

**Theorem 4.** *Function $tc_{COMMON}(h_1, h_2)$ does not return result for every tuple of harmonies $h_1$, $h_2$, even when it comes to harmonies with $|h| = 4$.*

*Proof.* These two harmonies do not have common keys: $cc\sharp dd\sharp$ (possible keys: *C minor*, *F minor*, *G minor*), $c\sharp dd\sharp e$ (possible keys: *C♯ minor* or *D♭ minor*, *F♯ minor* or *G♭ minor*, *G♯ minor* or *A♭ minor*) □

This is a simple consequence of the fact, that it is not possible to perform diatonic modulation from every harmony to every other harmony. We are here in a deadlock situation, because we propose, that TSD distance model should not evaluate modulations (it indeed does not – $TSD_{BASIC}$ treats every transition as a diatonic movement and if the modulation occurs, it wouldn't notify it) and we propose different ways of evaluating modulation complexity in the following chapter. However, $TSD_{BASIC}$ still should output how far is the transition from being „difficult“ (similarly to chord distances concept), so leaving the transition complexity unevaluated would not be a good idea. However, remembering our *constructing* approach in evaluating the complexity, we may still come up with some ideas, where even for those tuples of harmonies we can find the way how to disassemble one and construct the other one.

We propose 3 ways of completing the $tc_{COMMON}$ definition for those tuples of harmonies that do not have common keys, in order to achieve completeness as well:

1. $tc_{LAZY}$ behaves like if the harmonies had common keys and simply return $c_{BASIC}(h_1) + c_{BASIC}(h_2)$.
2. $tc_{COMPLEX}$ performs the *roots* function for both harmonies once again, but uses a modified version of *roots* without the optimization to omit trivial root harmonies. Then there is only $O(|A|)$ tuples for which we still would not be able to find common keys (usually single tones and clusters, proof can be

found simply through trying all types of harmonies starting from $|h| = 1$, for $|h| = 3$ we would find out that they all have common keys). For these special harmonies we have different options, but the best would be following the same *constructing* algorithm as in the rest of the model – we allow an empty root harmony. That would let us find a common root harmony even for these tuples and the resulting model would have the attribute of completeness. The transition then literally is disassembling one harmony and building the other from „scratch“.

3. $tc_{CHROMATIC}$ is based on the idea of chromatic modulation – even though the common key can not be found (= diatonic modulation), by altering several tones of $h_1$, obtaining $h'_1$, we may find the common keys for $h'_1$ and $h_2$. The *constructing* approach then advices: $tc_{CHROMATIC}(h_1,h_2) = tc_{COMMON}(h_1,h'_1) + tc_{COMMON}(h'_1,h_2)$. The completeness of this approach has to be treated individually, because even chromatic modulation is not possible from every key, harmony tuple to every other key, harmony tuple.

Because we want to preserve the uniformity of the $TSD_{BASIC}$ model, we choose the $tc_{COMPLEX}$ as the supplementary function to $tc_{COMMON}$. From the computational perspective, finding more roots only adds more cycles in our loops, so the complexity does not rise asymptotically.

**Definition 16.** *$tc_{BASIC}(h_1,h_2)$ behaves as $tc_{COMMON}$ in case that $\pi_R(roots_{BASIC}(h_1)) \cap \pi_R(roots_{BASIC}(h_2) \neq 0$, and as $tc_{COMPLEX}$ otherwise.*

#### 5.2.6 Comparison to Chomsky hierarchy

We quickly and informally compare the $TSD_{BASIC}$ model to the grammars of Chomsky's hierarchy, so we don't leave any confusion in between them.

$TSD_{BASIC}$ model indeed works as a finite set of context-sensitive grammars, however, with a coordinator (that should be touring-complete). If the coordinator, has CSG for each key $k$ and each root $r$, then by performing the function $root(h)$, it chooses those grammars that have their representation in $root(h)$. Each such grammar has the step from its start symbol to $r$, in its terminal alphabet it has the tones of $r$ and all the tones from $A - k$, the rest of the tones would be non-terminal symbols. Then the non-terminal symbols have rules for changing into respective augmented or diminished terminals ($ALTER$ rule), and the sentential form is prepared with the „ADD" non-terminals that can change into respective tone non-terminals, depending on the contextual information. The coordinator lets each grammar make a derivation of the harmony and then collect the resulting lengths, and output the best one.

However, even though the coordinating „machinery" makes $TSD_{BASIC}$ seem like even much more complex system, we need to conclude with, that it of course can not derive any non-regular harmony, because of the finite array we work with ($\mathbb{Z}_{12}$, maximum of $2^{12}$ harmonies). Even if we broaden the set of tones to all audible pitches, we still get only finite number of pitches and therefore also the harmonies. The grammars are therefore here only for the „feel" of working with a known model.

We may also conclude that the space of all musical pieces we analyze is regular, since it is a sequence of harmonies and we accept it by our program, stepping from one to another, in a process equivalent to DFA. In the next section 5.3 we give the reader an insight on how the transition function can look like, and show a nice graph representation of our model.

## 5.3 Graph representation – Christmas tree model

A beautiful aspect of our model is, that it is finite – since the harmony universe contains precisely $2^{12}$ harmonies. So another beautiful aspect is, that the results for $tc_{BASIC}$ can be pre-calculated, simply by running the function $\frac{2^{12}.(2^{12}-1)}{2}$ times, since $tc_{BASIC}(h_1, h_2) = tc_{BASIC}(h_2, h_1)$. This yields to the creation of a graph that represents our model.

However, the resulting graph with weighted edges would be complete, therefore huge and very impractical to store and create, even if we bound the number of tones by a constant smaller than 12. Therefore we conclude our description of $TSD_{BASIC}$ model with another graph representation, much more attainable. Snippets of it we have already used in the previous section. Due to its meaningful appearance we call it **Christmas tree model**.

**Christmas tree model** is a graph representation of $TSD_{BASIC}$ model for a specific key, $CTM(k) = (V,E)$, where the root node represents family of harmonies – all possible root harmonies for a given key, and all the other nodes represent a single harmony. The edges represent either an rule ADD or ALTER and each edge has a weight 1.

The basic form of Christmas tree model can be seen on figure 18a. The main harmony functions listed on the top represent all the roots (*T* = possible tonic roots, *S* = possible subdominant roots, *D* = possible dominant roots) and the triangular clusters represent the graphs that are created by modifying the root, as already seen e.g. on figure 17. As we have denoted earlier, the arrows in between $T$, *S*, *D* are „zero“ edges, because they do not hold any transition complexity. By definition, therefore, we should merge them into one node. However, the same applies also for some nodes in the clusters – many harmonies can be derived from multiple functions. For example the harmony $cefa$ can be derived in *C major* from root harmony *ce* as a Tonic modification, but also from the root harmony $cfa$ which is a Subdominant. Hence we modify („decorate“) a tree with markings, that make clear where the nodes are merged into one, see figure 18b.

It is important to notice, that merging the nodes destroys the tree structure. In this merging, of course, lies the functionality of our model (it is the sound of the harmony that matters, plus we are interested in the distance from the whole *T* – *S* – *D* progression and let the functions rename dynamically). But because it also destroys the *Christmas tree* representation, we prefer using the above figures as

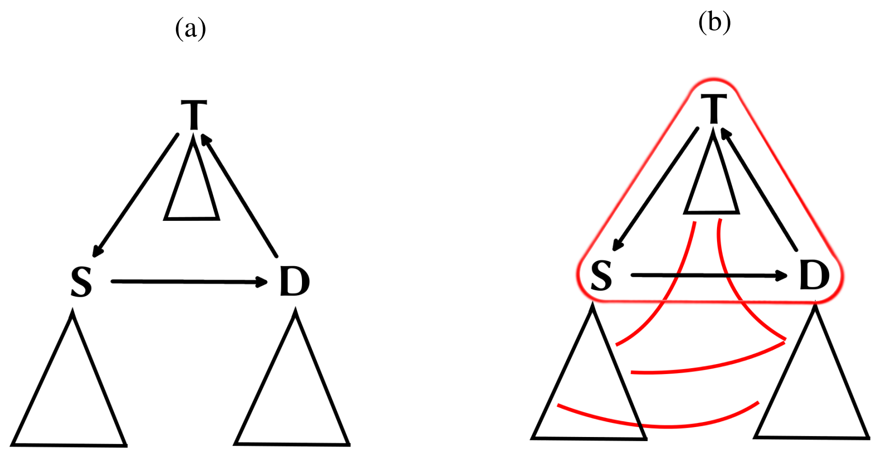



Figure 18: Christmas tree model in its basic form (a); and including zero edges (b)

the visual representation.

The next theorem summarizes what the merging of the nodes really does in our model.

**Theorem 5.** *By merging two nodes (combining them into one and deleting the created loop from the node to itself) in the graph containing only edges with weight 1, we again obtain the graph containing only the edges with weight 1.*

*Proof.* Since both of the nodes did have only the neighbours in a distance 1, and now they are forming one node, their neighbours combine together and are again in a weighted distance 1 from the new node. □

Although we loose the possibility to use some tree searching algorithms, we are given a solid graph with equal edges where each harmony is represented by one node with the exception of root nodes. The next section explains a little bit more the significance of the rood nodes.

### 5.3.1 Christmas forest

Spreading the idea towards all of the keys, we get the graph that we may call *Christmas forest*, see figure 19. Starting from root nodes of 24 major and minor keys, The edges intertwine together heavily towards the more complex harmonies, having many common nodes (on figure coloured) – in fact, the pivot chords of possible modulations. Note that, we did not mark the possibility of traversing directly between the basic harmony functions which is possible – between the keys that are in the relationship of perfect fifth (from tonic *C major* directly to tonic of *G major* since it is the zero edge in *C major* from *T* to *D*). We did it on purpose – imagining that there are zero edges as well between the colorful triangles from the figure and that we can therefore „travel "on the whole circle of fifths for free leads to a misunderstanding of the concept. The root nodes form one node, but still contain multiple harmonies in which we can not be at one time. We can explain it the best by the following game example. We suggest the reader to follow it along with looking at the figure 19:

**Example 4.** *Super Mario is running through the Christmas forest, searching for his blond princess. While he searches, it is difficult (complex) to move and he loses one point for every edge he travels over (optionally the computer-fashioned music also plays the harmonies along). Sometimes he gets tired and looks for the triangular colorful house to recover. Often he finds himself quite near one of them. When in the house, it is not difficult (complex) to move anymore, but it still takes some time. The triangular house has 3 main compartments and each of them a room for a giant (three tones), 2 human sized (two tones) and one lilliputian person (one tone, naturally, Super Mario fits to all of them). The only thing he notices is the light flashing in the room when he is in, while in the other rooms there is dark. Super Mario is interested by that flashing, and he finds a hidden teleport in the room, and if he chooses, suddenly he can reappear in a house with different color! Not to mention, that the teleporting does not take the time at all. Later he realizes, that every time he enters the house and the room in the house, particular other rooms in other houses (in the distance of the fifth) flash the light*

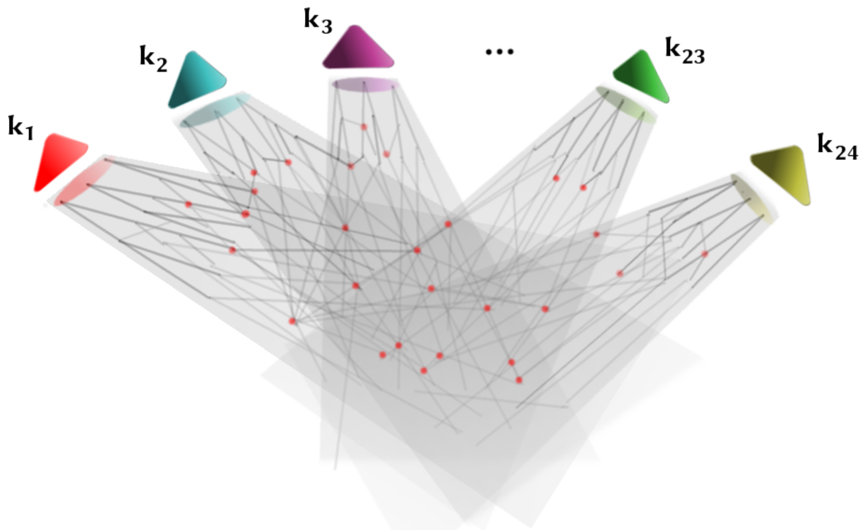



Figure 19: Christmas forest

*too – a teleport being activated.*

Therefore, we need to differentiate, that even though we can get to the other root harmonies without any complexity (teleport), we can do it in one step only if these root harmonies are in Tonic, Subdominant or Dominant relation to the other root harmony. To get to some other root harmony we need another step. This also illustrates the point, that we indeed do not track modulations and it should be done separately.

## 5.4 On the computational complexity of the model

The importance of the Christmas tree model lies in the theoretical possibility of implementing the $TSD_{BASIC}$ model by pre-calculating the graph prior to analysis. Such pre-calculation would still contain all harmonies from $U$, therefore bounding the total number of harmonies might be a good idea, for example by bounding the maximal number of tones. However, the creation is quite fast – we can inductively generate all the harmonies starting from all the roots and using the rules *ADD* and $ALTER$ and remember only the trivial edges, as opposed from the complete graph at the beginning. Then, having the two harmonies $h_1$, $h_2$, the speed of algorithm would only depend on the graph algorithms. However, we can also use the algorithms built around the pseudocodes provided in previous chapter and not on the graph creation and search. In this section, we consider both ways of implementation and evaluate their computational complexity.

### 5.4.1 Time complexity of the main functions

Let $n$ be the maximal size of the harmonies used ($n = 12$ if not set differently), $k$ the number of keys used ($k = 24$), $f$ the number of functions within $k$ ($f = 3$). From the analysis of the pseudocode we get:

- $roots_{BASIC}(h) \in O(kfn)$
- $c_{BASIC}(h) \in O(k^2fn)$; if we work with the search of the *root* database, we use the upper bound for the keys $k$ and the upper bound for the roots $kf$
- $tc_{BASIC}(h_1, h_2) \in O(k^2fn^2)$

More optimizations are possible, mostly by caching the expensive look-ups in the database for keys and functions, but since these are constant, then quadratic complexity from the length of harmony is not bad for the analysis. If the performance of the algorithm would be slow, we may lower the maximal size of harmonies, thus lowering the quadratic element.

If we choose the graph traversal algorithms, the best option is to use breadth-first search (BFS), since the costs of the edges are all equal to 1, resulting in complexity $O(2^n)$ because the amount of vertices is $2^n$. Even though exponential, comparing the pseudocode approach ($24^2 * 3 * 12^2 = 248832$) and graph search ($2^{12} = 4096$) yields to the usage of the graph algorithms.

## 5.5 Evaluating the complexity of the musical piece

Finally, by using the $TSD_{BASIC}$ model on the chord sequence $\{C_i\}_{i\leq l}$ (which is the sequence of harmonies), we get the final harmonic complexity of the piece. There are many options on how to do it, moreover we have *transition complexity* (as a sequence $\{t_i\}_{i<l}; t_i = tc_{BASIC}(C_i, C_{i+1})$), but also *harmony complexity* (as a sequence $\{h_i\}_{i\leq l}; h_i = c_{BASIC}(C_i)$) that we can both use.

Also, we have mentioned earlier the analogy with computational time complexity – and in fact, our model implements this idea literally, $tc$ outputs exactly the *assembling* or *disassembling* time, the work needed to do to change one harmony to another. We remain true to this analogy too.

Some data might help first: The output of one call of $tc$ is in $< 0, 44 >$ in theoretical perspective, because starting from 0 tones, we can add 12 tones, alter 5 of them and add 5 new tones harmony complexity, times 2 for upper bound of transition complexity. Normally, the values are somewhere between 0 and 10, 0 - 5 we might encounter commonly in classical and popular music, 10 is already for chords with 3 or more added dissonances and clusters (note that, this is taken from notation, these values may change based on the threshold we use and for the audio are usually higher).

It comes naturally, that we should be interested in some absolute numbers. We provide the following definition.

**Definition 17.** *For musical piece M, sequence of its transition complexities* $\{t_i\}_{i\leq l}$ *and harmony complexities* $\{h_i\}_{i\leq l}$*, we define the following complexity measures:*

- ***Average transition complexity****:* $ATC_{model}(M) = \frac{\sum_{i=0}^{l-1} t_i}{l-1}$
- ***Maximal transition complexity****:* $MTC_{model}(M) = \max(t_i)$
- ***Average harmony complexity****:* $AHC_{model}(M) = \frac{\sum_{i=0}^{l} h_i}{l}$

- ***Maximal harmony complexity**: $MHC_{model}(M) = \max(h_i)$*

The computational time complexity might come in handy if we want to obtain a relative measure. We can indeed calculate the assembling time based on the input – which is for one transition the first harmony of the transition. As in the complexity theory, we simply compare the length of the input to the final time of execution. This idea is indeed great to differentiate those musical pieces that contain one or two voices from the pieces with more full harmonies – if the analysis not based on input length outputs that they have the same complexity, the listener can perceive it differently and find the first one very complex, because the dissonances were more *audible* or *disturbing*.

Unluckily, there are some differences. Even though it works in the way that – if we only perform 1 rule no matter how long the harmony is, we get constant complexity; if we always perform 1 rule with every tone, we get linear complexity – however, we may also get zero complexity and we can never get quadratic or higher complexity. In other words, the fact, how many times we perform a rule on the tone doesn't depend on the input at all, it is either 0, 1 or 2.

Nevertheless, we use this idea to obtain one more measure:

**Definition 18.** *For musical piece M, sequence of its transition complexities $\{t_i\}$ and sequence of its harmonies $\{C_i\}$, we define **relative transition complexity**:*

$$RTC_{model}(M) = \frac{\sum_{i=0}^{l-1} t_i}{\sum_{i=0}^{l-1} |C_i|}$$

*where $|C_i|$ is the size of i-th harmony from the sequence $\{C_i\}$.*

### 5.5.1 Time complexity of the music analysis

Secondly we would be interested in the time complexity of analysis of the musical piece. Although the concrete implementations might differ, we base our estimations on the figures of Harmanal system from chapter 3. We do not take the feature extractions algorithms into our analysis, nor the smoothing 1 since smoothing 1 depends on the outputs of the Vamp plugins. We start from obtaining beat-synchronized chromas. Let $l$ be the number of beats from the audio and $k$, $f$, $n$ are used as defined in the section 5.4.1.

- We do $O(ln)$ operations to get the chord candidates, since we only compare the chroma features to the threshold
- We do $O(l)$ algorithms for smoothing 2, thus obtaining the chord sequence $\{C_i\}$, with the length bounded by $l$
- We then do $l$ times calculation of transition complexity, resulting in complexity $O(k^2 f n^2 l)$ or $O(2^n l)$ depending on which implementation we choose for the model
- Finally we calculate the harmonic complexity of the piece in $O(l)$. We also in $O(l)$ revise the list of potential labels for the harmonies using convolution method

Since $n$, $k$, $f$ can be considered constants, our complexity analysis is therefore $O(l)$.

# 6 Harmanal application

In this section we provide the implementation details and a quick guide through the Harmanal application. The requirements for the application can be seen in section 1.3 and the outlining diagrams of the system in section 4.1.

## 6.1 Technical information

| Harmanal, version 1.0, May 2013 | |
|---|---|
| type | Java application |
| platforms | Linux, Windows, Mac OS, Java applet |
| licence | GNU GPL |
| dependencies | JRE 6 or higher (http://java.com/en/download/)<br>NNLS Chroma and Chordino Vamp plugin 0.2.1 or higher (http://isophonics.net/nnls-chroma/)<br>QM Vamp plugin set 1.7 or higher (http://vamp-plugins.org/) |
| documentation | http://www.riesky.sk/~laco/web/harmanal/documentation/ |
| download | http://www.riesky.sk/~laco/web/harmanal/download/ |
| system components | Chordanal 1.2;<br>NNLS Chroma and Chordino plugin 0.2.1;<br>Bar and Beat tracker plugin 1.7;<br>JVamp 1.2;<br>JNA 3.5.2 |

Table 6: Harmanal - technical information

## 6.2 Overview

Harmanal application lets the user do harmony analysis - chord transcription from audio, chordal analysis from MIDI input devices and harmonic complexity evaluation – all in one place.

It is divided into 2 tabbed windows:

- **Chord transition tool**
  - *Input:* User chooses MIDI input device or text fields to input two harmonies. Common virtual keyboard applications are supported too.
  - *Outputs:* Several outputs are provided
    - Names of the harmonies
    - Relative structures
    - Keys
    - Functions
    - Root harmonies (as defined in section 5.2.2)
    - Harmony complexities (as defined in section 5.2.3)
    - Transition details
    - Transition complexity (as defined in section 5.2.5)
- **Audio analysis tool**
  - *Input:* User chooses a WAV file for analysis
  - *Outputs:*
    - ATC - Average Transition Complexity (as defined in section 5.5)
    - MTC - Maximal Transition Complexity (as defined in section 5.5)
    - AHC - Average Harmony Complexity (as defined in section 5.5)
    - MHC - Maximal Harmony Complexity (as defined in section 5.5)
    - RTC - Relative Transition Complexity (as defined in section 5.5)
    - Chroma features (txt file)
    - Chord sequence (txt file)
    - Transition complexities (txt file)

Version 1.0 of the application is included in this work. For the latest version, please visit: http://www.riesky.sk/~laco/web/harmanal/

## 6.3 Implementation details

Harmanal application takes full advantage of Java object oriented environment, decomposed into comprehensible subsystems, and flexible for future extensions.

The main system components are: **Harmanal**, **Chordanal**, **Application GUI**, **Database**, **MidiHandler**, **NNLSPlugin**, **BeatTrackerPlugin**, **Testing environment** and a comprehensive system of music classes coming with Chordanal, introduced in [13].

### 6.3.1 Harmanal static class

**Harmanal** is a static class in Harmanal system responsible for all the tonal harmony and harmonic complexity related events: *grammar derivation, key finding, root harmonies finding, complexity evaluation.*

In version 1.0 Harmanal static class is implemented to make look-ups to the Database for key-related information and to simulate grammar derivation each time when asked for transition complexity. As proposed in computational complexity analysis in section 5.4.1, in future versions a Christmas tree model generation and graph search algorithms can be introduced to provide faster, even real-time, outputs.

### 6.3.2 Chordanal static class

**Chordanal** is a static class in Harmanal system responsible for all the naming and structure analysis related events: *factory methods to create music entities, naming methods, abbreviating methods, parsing methods, music entities analysis*

Chordanal was first introduced in 2010 in [13]. Its powerful naming and parsing capabilities for chords used for Ear training were re-used in this work. Some advanced terms might be still missing in 1.2 due to the translation from Slovak language. Chordanal's strength lies in the look-ups to the Database full of data from music theory, that it is able to recreate on every run of the program.

Along with Chordanal static class, an object oriented framework for music entities have been introduced. Following classes are contained in version 1.2: **Tone**, **Harmony**, **Key**

### 6.3.3 Application GUI

For GUI documentation, visit:
http://www.riesky.sk/~laco/web/harmanal/documentation

### 6.3.4 Other components

**MidiHandler** is a class responsible for any MIDI related events: thanks to Java Sound API it is able to catch MIDI events as well as send MIDI signals to play tones.
**NNLSPlugin** developed by Mauch [14] and **BeatTrackerPlugin** developed by Stark and Davies [22] are Vamp plugins developed under GNU GPL licence that were integrated into Harmanal using JVamp wrappers for native C++ Vamp plugins and JNA library. Provided that the user installs the plugins on his machine, JRE is able to load the respective libraries and Harmanal can run cross-platform.

## 6.4 Screenshots of usage

When user runs the Harmanal application, a tabbed window with *Chord transition tool* is opened so he can start his queries right away. The usage is intuitive – from up to down, first the user selects from the available MIDI devices. If everything works fine, next the user sees that it is possible either to press the „Capture“ button or to use a text field. When the capture button is on, all played MIDI signals are being processed and as soon as the button is off, the user sees the analyzed input in most of the text fields. As soon as he inputs the second harmony, the rest of the text fields containing the transition information are filled out. A nice feature is, that if the user is not happy with the input, he or she may modify or reassign the input using textfield – the easiest way is using the relative text field, where he or she simply writes e.g. *C E G* to get the *C major chord*. The screenshot of the usage of *Chord transition tool* is on figure 20.

When the user wants to analyze audio files, he or she selects the other tab with *Audio analysis tool* label. Again, the usage is very straightforward: from up to down. Importantly, the user must first hit the button „Load plugins“– it usually takes around 1-2 seconds to load the plugins. The user is notified if there was a problem in loading the plugins. Then the user inputs the URL of the WAV he or she wants to analyze, optionally changes the output txt files and hits the button „Analyze“. Normally, it takes around 10-15 seconds for an analysis of 3 minute WAV file, 44100 Hz, 16bit samples. When done, the user reviews the filled text fields and may open the files for further analysis information. The screenshot of *Audio analysis tool* is on figure 21.

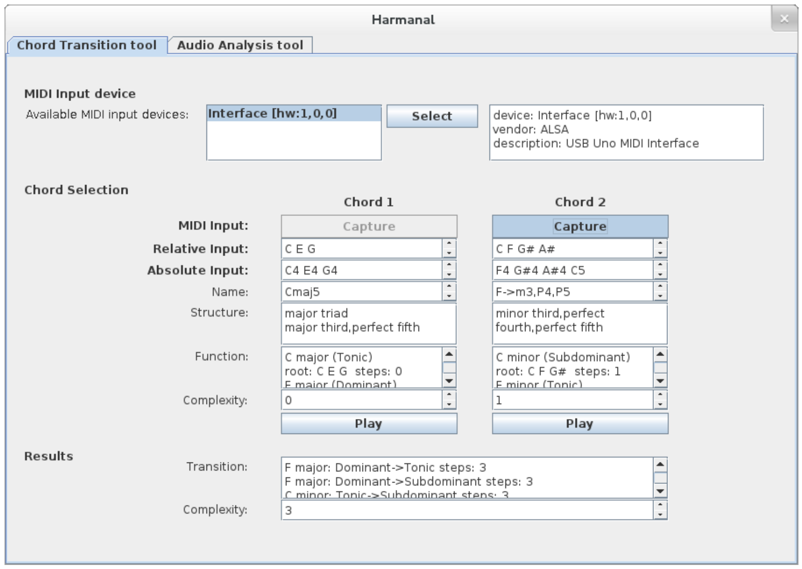



Figure 20: Harmanal application - Chord transition tool

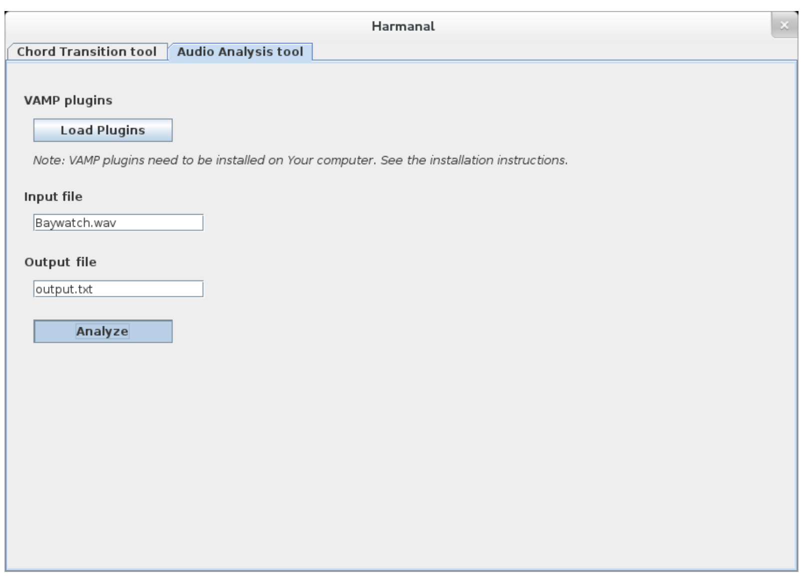



Figure 21: Harmanal application - Audio analysis tool

# 7 Results of analysis

Harmanal system was tested on the musical pieces from:

- Top 5 Best-selling Rock bands[18]: *The Beatles, Led Zeppelin, AC/DC, The Rolling Stones, Queen*
- Top 3 Best-selling Pop artists: *Elvis Presley, Michael Jackson, Madonna*, and leaders of Pop music charts in 2013: *Rihanna, Bruno Mars*
- Top Jazz artist[19] *Theresa Andersson*, other Jazz artists: *Hiromi, The Jazz Invaders*
- Significant composers of classical music from different periods
  - Rennaisance: *G. Allegri, G.P. da Palestrina*
  - Baroque: *J.S. Bach, G.F. Handel, T.G.Albinoni*
  - Classicism: *W.A. Mozart, L.v. Beethoven, L. Boccherini*
  - Romanticism: *P.I. Tchaikovsky, H. Berlioz, E.H. Grieg*
  - 20th century: *G. Gerschwin, I. Stravinsky, A. Honneger*
- Other artists from different genres (Rap, Folk music, etc.)

For each artist an average of 5 titles were analysed. The aim was:

1. To compare the **genres** and **periods** against each other
2. To compare the **artists** amongst the genres
3. To compare the **songs** from the same artists and find any significant deviations

Meanwhile, the purpose of our experiments was mainly:

[18]Source: http://www.billboard.com
[19]Source: http://www.artistsdirect.com

- To see whether the our system behaves in the similar way that the skilled musician would analyze and see the usefulness towards recommender systems

- To gather feedback based on the results, for further improvements

As a bonus, several challenges were proposed (Queen vs Classical music, Rock bands competition, etc.).

For the following analysis, we have used a version of Harmanal system optimized for good performance and more precise results. Most importantly, Chordino plugin version 0.2.1 by Mauch [14] was used instead of Bar and Beat tracker plugin for finding the locations of harmony changes. Using Chordino plugin already set up for finding where the harmony significantly changes and averaging the chromas according to given locations yielded better results than averaging the chromas for every beat, because it helps to locate only the significant changes. Chordino plugin is a GNU GPL licenced Vamp plugin that comes in a package with NNLS Chroma plugin[20]. Amongst other optimization we have chosen to bound the *maximal size of harmonies* and set the correct *threshold*[21], as defined in sections 4.1.

[20]http://isophonics.net/nnls-chroma/

[21]The Harmanal system was calibrated with the *threshold* value $T$ of 0.05, so only chroma features above the threshold value are considered. We have set the *maximal size of harmonies* $n$ to 4 to speed up the analysis. If the model can not find the diatonic transition between the two harmonies, we have used simplified version of the $tc_{COMPLEX}$ function, which only assigns the maximal transition complexity of 7 to these harmonies, since according to previous experiments 7 was the highest transition complexity possible due to our calibration. For more information on transition complexity, see section 5.2.5.

## 7.1 Comparing genres and historical periods

We first focus on a basic task – comparing the different genres of music, and for classical music, the different historical periods. The results can be seen in tables 7 and 8 and on figure 22. Since *Average transition complexity* (see section 5.5) provided the best results from all the proposed values, we chose to include this value into our final visualizations.

| | **Jazz** | **Rock** | **Pop** |
|---|---|---|---|
| **ATC** | **3.148** | **2.249** | **1.981** |
| MTC | 7 | 5.842 | 5.636 |

Table 7: Comparing music genres

| | **20th century** | **Romanticism** | **Classicism** | **Baroque** | **Rennaisance** |
|---|---|---|---|---|---|
| **ATC** | **3.512** | **2.713** | **2.195** | **2.012** | **1.424** |
| MTC | 7 | 7 | 6.667 | 6 | 7 |

Table 8: Comparing historical periods in classical music

(a) (b)

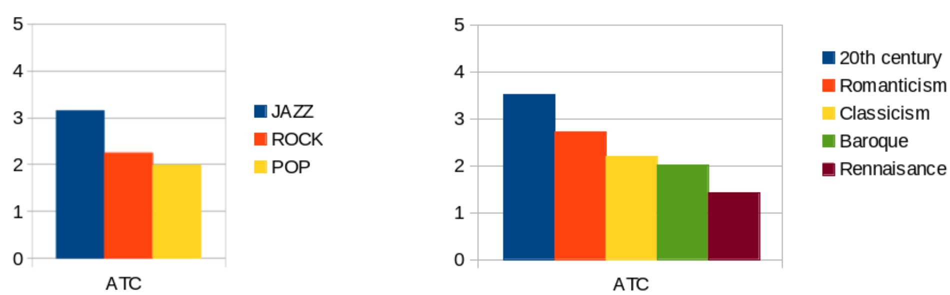



Figure 22: Chart for comparing music genres (a) and historical periods in classical music (b)

## 7.2 Comparing artists and titles

Next we have selected the Rock genre to see if there are differences in harmonic complexity in between the artists. The results can be seen in table 9 and on figure 23.

| | Queen | The Beatles | The Rolling Stones | Led Zeppelin | AC/DC |
|---|---|---|---|---|---|
| **ATC** | **2.469** | **2.131** | **2.094** | **2.087** | **1.915** |
| MTC | 6.833 | 4.833 | 5 | 5.667 | 5 |

Table 9: Comparing rock bands

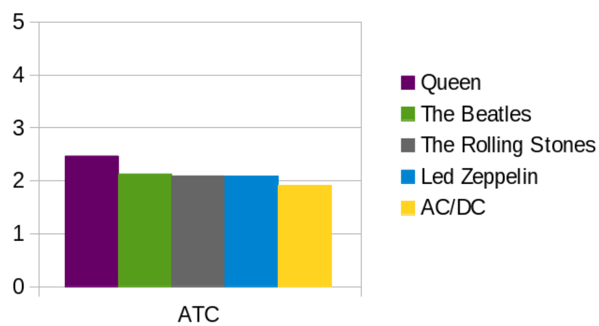



Figure 23: Chart for comparing rock bands

Another aspect we were interested in was whether some songs from a certain artist deviates from their usual production. The results for a sample of titles from Queen and The Beatles can be seen in tables on the figure 24 and in chart on figure 25

(a)

| Title | ATC |
|---|---|
| We Are The Champions | 2 |
| Don't Stop Me Now | 2.352 |
| Bicycle Race | 2.882 |
| *Radio Ga Ga* | *1.604* |
| Bohemian Rhapsody | 2.576 |
| Crazy Little Thing Called Love | 3.4 |

(b)

| Title | ATC |
|---|---|
| A Hard Day's Night | 2.276 |
| Hey Jude | 1.414 |
| Michelle | 3 |
| Twist And Shout | 2.204 |
| Yellow Submarine | 2.554 |
| *Let It Be* | *1.336* |

Figure 24: Analysis of Queen (a) and The Beatles (b) songs

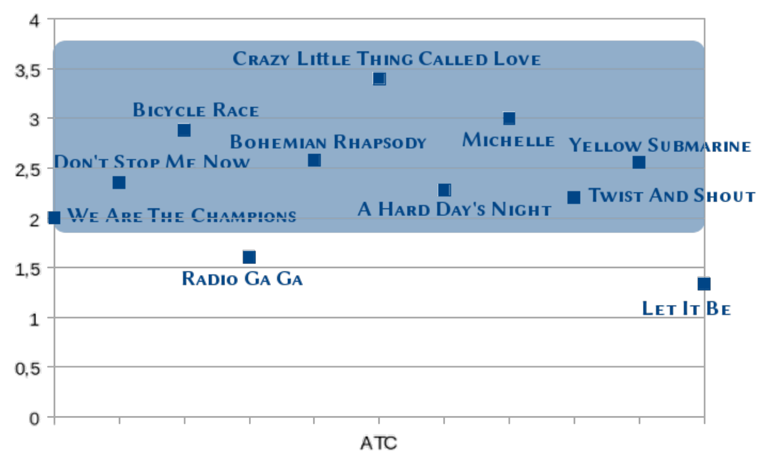



Figure 25: Song comparison of Queen and The Beatles songs

## 7.3 Other sample results

We provide a table of other sample results for illustration, sort by the music genres.

| | |
|---|---|
| **artist:** | **Queen** |
| **title:** | **Bicycle Race** |
| **ATC:** | **2.882** |
| AHC: | 1.036 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 0.758 |

| | |
|---|---|
| **artist:** | **The Beatles** |
| **title:** | **A Hard Day's Night** |
| **ATC:** | **2.276** |
| AHC: | 1.013 |
| MTC: | 5 |
| MHC: | 3 |
| RTC: | 0.588 |

| | |
|---|---|
| **artist:** | **AC/DC** |
| **title:** | **Highway To Hell** |
| **ATC:** | **1.915** |
| AHC: | 1.000 |
| MTC: | 5 |
| MHC: | 2 |
| RTC: | 0.492 |

Figure 26: Sample results for the Rock music

| | |
|---|---|
| **artist:** | **Michael Jackson** |
| **title:** | **Billie Jean** |
| **ATC:** | **1.812** |
| AHC: | 1.117 |
| MTC: | 4 |
| MHC: | 2 |
| RTC: | 0.452 |

| | |
|---|---|
| **artist:** | **Bruno Mars** |
| **title:** | **Just The Way You Are** |
| **ATC:** | **1.642** |
| AHC: | 1.019 |
| MTC: | 4 |
| MHC: | 2 |
| RTC: | 0.426 |

| | |
|---|---|
| **artist:** | **Jimi Jamison** |
| **title:** | **Baywatch Theme (I'll Be Ready)** |
| **ATC:** | **1.543** |
| AHC: | 0.914 |
| MTC: | 4 |
| MHC: | 2 |
| RTC: | 0.401 |

Figure 27: Sample results for the Pop music

| | |
|---|---|
| **artist:** | **Hiromi** |
| **title:** | **010101 (Binary System)** |
| **ATC:** | **4.242** |
| AHC: | 0.801 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 1.35 |

| | |
|---|---|
| **artist:** | **The Jazz Invaders** |
| **title:** | **Licks And Brains** |
| **ATC:** | **2.622** |
| AHC: | 1.304 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 0.666 |

| | |
|---|---|
| **artist:** | **Theresa Andersson** |
| **title:** | **Birds Fly Away** |
| **ATC:** | **2.581** |
| AHC: | 1.279 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 0.653 |

Figure 28: Sample results for the Jazz music

| | |
|---|---|
| **artist:** | **Arthur Honneger** |
| **title:** | **Pacific 231** |
| **ATC:** | **4.346** |
| AHC: | 1.684 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 1.096 |

| | |
|---|---|
| **artist:** | **Ludvig van Beethoven** |
| **title:** | **5th Symphony: 1.Allegro con brio** |
| **ATC:** | **2.633** |
| AHC: | 0.953 |
| MTC: | 7 |
| MHC: | 3 |
| RTC: | 0.716 |

| | |
|---|---|
| **artist:** | **Gregorio Allegri** |
| **title:** | **Miserere Mei Deus** |
| **ATC:** | **1.462** |
| AHC: | 0.418 |
| MTC: | 7 |
| MHC: | 2 |
| RTC: | 0.44 |

Figure 29: Sample results for the Classical music

## 7.4 Results summary

The results of our analysis have proved that our system is useful tool for music analysis. We illustrate this by the following summary, which includes also other results not present in the previous tables. First of all, we can see that the the complexity values do not deviate from an expected analysis made by the skilled musician in these points:

- Jazz is the most complex genre, followed by rock and pop genres.
- 20th century classical music and Music Romanticism lead the charts in classical music analysis.
- The leader of the rock artist complexity chart, music band *Queen* is famous for their choir passages and interesting harmonies
- As far as concrete songs, high ratings of Queen songs *Bicycle Race* and *Bohemian Rhapsody*, or *Michelle* by *The Beatles* can be understood by anyone who knows how harmonies change in these songs. Even more understandable are the absolute complexity leaders:
  - *Hiromi* with her piece *Binary System*, a jazz piece, that was the closest to computer music from what we've had ($ATC = 4.242$) in our analysis.
  - In classical music it is *Arthur Honneger* with his piece *Pacific 231* ($ATC = 4.346$).

As we have mentioned earlier, the $ATC$ values proved to be the most useful, with values ranging from 0 to 5. Even though the range is still small, one must understand that it is the computational time of constructing the harmonies which is bounded, depending on what maximal size of harmonies we use. $MTC$ has provided us with results stating that there has been at least one very complex transition in the piece (maximum of $MTC = 7$ was used by our model). However, such result might be misleading, because sometimes this can happen due to the unprecise tuning of the WAV (probably the case also in analysis of Rennaisance pieces).

During our analysis we have also found other results, that prove that our system does what it is supposed to do, but we might not expect them from the beginning:

- High rating of Jazz is caused by using many chromatic movements. The TSD distance model of course has trouble finding diatonic solutions to these movements and needs to output high transition complexities. Such behaviour can be also clearly seen amongst the Queen titles, where the highest score is scored by the song *Crazy Little Thing Called Love* which has very close to Jazz music.

- Although modern popular music is scoring ATC values almost always below 2, we can conclude that Rap music can have on the other hand very high values (*Eminem*: *Lose Yourself*; $ATC = 3.032$). This is caused by the spoken words being in the dissonance with the music accompaniment. This behaviour can be also seen e.g. on high ratings of *Yellow Submarine* by The Beatles, which has passages in which spoken word may be heard.

- The same applies for songs with very colorful instrumentation, ornamental singing, or even Pop music using where synthetic or electronic instruments deviate from the standard pitch classes, e.g. by continuously changing the frequency. An example of a song where all of these things apply along with multi-voice singing and rap is *Black Eyed Peas*: *Let's Get It Started*; $ATC = 3.642$.

Even the results that were not expected show usefulness of our model. Different aspects of music like intrumentation, choirs, and using different harmonies form a music *style* of an artist and our model can be capable of detecting such style. One example, which should be however proved by more experiments is, that another rock band, *Pink Floyd* has a style that would rate even higher than Queen in our analysis (*Goodbye Blue Sky*; $ATC = 2.778$).

Interesting results were shown also by analysis searching for title that deviates from the usual production of the rock bands:

- The title *Radio Ga Ga* by Queen is known by more popular character than the rest of Queen songs, and therefore resulting lower complexity.
- The title *Let It Be* by The Beatles has quite simple chord progression, which can be played by amateurs on guitars, and was also found by our application. to have lower complexity

Last but not the least, we have also shown an interesting the way how to compare the musical pieces from different genre. One example is, that on the figure 24 for rock songs of *Queen* and *The Beatles*, the songs that are in the upper part of the blue rectangle would harmonically fall to Romanticism in classical music. On the other hand, the lower part of the blue rectangle has results similar to Classicim in classical music.

All of these results show that harmonic complexity can be used effectively for Music Information Retrieval tasks. Moreover, the results guide us to further improvements on TSD distance models, but not only there – future works can be done on describing other complexities complementary to TSD distance, With these complexities we could gather more precise results, and we discuss them in section 8 dedicated to future travels.

# 8 Future works

In this section we provide an overview on what other harmonic complexities are there other than TSD distance complexity, to give as complete picture as possible and give the ideas for future works.

## 8.1 Five harmonic complexities

There are different views on complexity when it comes to harmony, even from the theoretical perspective (not talking about music perception or machine learning). It can be either how simple or complex transitions are being used (chord distance concept, as we have described in the chapter 5, it is close to the computational complexity from complexity theory). It can be how often the repetitions are being used — refrain, verses, etc. (space complexity). Or, how *fast* the transitions appear (speed of transitions, this resembles computational time complexity too).

Then in transitions, we might differentiate, whether any modulations were used in the piece, how often and between what keys, because majority of chord distances would not take it into consideration. And there are also another 2 ways how to look at the transitions between the harmonies — instead of evaluating the simplicity or complexity of the transition, we can also find how far they are from tonic (tonal tension) or, how smooth are the transitions (voice leading concept).

To summarize – five complexities in music harmony:

1. chord distances and tonal tension
2. voice leading
3. modulations
4. repetitions
5. transition speed

In this thesis, we have provided the proposal for the first one, that takes both chord distance and T – S – D rules into account, We shortly summarize some proposals for the other ones as well.

These complexities may be better perceived as categories – we put chord distances and tonal tension together, because they both take care of the transitions between the chords, only evaluate them from the different perspective. Tonal tension uses rather cumulative approach for the musical piece, whereas chord distance suffices with 2 chords. Our approach from chapter 5 combined these two approaches together. We can say that in general, complexity algorithms can either focus on the full category, or on some concrete subcategories.

Some of the algorithms may combine multiple categories into one, for example Lerdahl's chord distance is taking also the key relationships into account [9]. However, to provide more verbose and accurate output, we propose to leave the categories separate. All five categories work on different levels. For example, $1^{st}$ and $2^{nd}$ and $3^{rd}$ would be from different levels of tonal pitch space: $1^{st}$ is how triads change within the key, $3^{rd}$ studies relationships between the keys, and $2^{nd}$ studies the tones within the triad. So we propose that the resulting harmonic complexity for a musical piece would then use five different scores. We consider it a good practice, as opposed to obtaining only one number/score – nobody would understand how we calculated that number. Of course, some heuristics can be used to calculate the total score as well.

### 8.1.1 Voice leading complexity

This is a proposal for calculating the harmonic complexity from a different perspective – voice leading. Both tonal harmony and theory of counterpoint rules can, and should, be used. The idea that may be thought of for future works is, whether a *LEADING* rule can be safely introduced, in the same fashion that *ADD* and *ALTER* rules. Such *LEADING* rule should however, more like generative system (g-systems) rather than grammars, be used on every tone of the harmony. Instead of the complexity of the overall transition we would get the complexity of the parallel voice leading.

Different possibilities occur: Moving semitone up or down (leading tone) may be considered as not complex, whole tone can be considered more complex. Perhaps it should be distinguished whether the movement was within the key, or outside the key, or the complex rules of tonal harmony can aid us in deciding what should happen.

### 8.1.2 Complexity of modulations

There have been several attempts on describing the modulations and we believe that soon enough also some methods arise to compare all of types of modulations and evaluate their complexity. The best practice seems to be modulations evaluation based on number of steps on the circle of fifths.

### 8.1.3 Space complexity

As an important note on our TSD distance complexity, we provide an example: A song or classical piece that would use 3, very harmonically interesting transitions, may appear very interesting at the beginning. But as soon as it would rotate these 3 chords all the way to the end, we would loose interest quickly. Space complexity needs to be therefore considered in the final evaluation of complexities. We propose using pattern matching methods, to find all similar regions e.g. through self-similarity matrices, or more easily, using the extracted chord sequence. Then the resulting complexity is the total length of transitions that are not repeated anywhere in the piece. This approach is similar to Knuth [7].

### 8.1.4 Transition speed

Transition speed simply denotes how many transitions per unit of time are used. Simply stated, however this needs more sophisticated algorithm to find the chord boundaries, because approximating transitions on beats would lead to imprecise results. We refer to [16] for more information on chord segmentation.

# 9 Conclusion

In this work we have presented a new model for evaluating harmonic complexity of musical pieces. Our work was focusing on chord transition complexity, which is amongst the five proposed complexities from section 8.1 one most important and studied. We have formally and programatically described the grammar-based model and we have given proofs of its completeness. We have provided the graph analogy and visualization called Christmas tree model to better understand the model and optimize its performance. Since harmonic complexity is a new term, we have also defined the measures for the musical piece that can be researched. Lastly we have analyzed the asymptotic computational complexity.

We have also provided three added values – one is an written overview of the interesting world of music theory and music information retrieval, from where it all begins up to current MIR research, comprehensible even for a non-musician, to encourage young researchers that might be interested in this field. Second is implementing a multi-platform application Harmanal capable of complexity analysis. And the third is the set of experiments we have done on more than 70 songs to show the comparison of different genres and artists. The aim was not only to provide interesting information, but also to help our model become helpful for future, for even more practical usage.

For future works, much can be done in specifying the other proposed complexities, or improving our model. Moreover, there still is the motivation we have described in the introduction – implementing a recommender system based on music complexity.

Lot of other models *around* harmonic complexity have been proposed and studied, such as chord distance or tonal tension, but the term music complexity was mostly omitted in formal conversations, perhaps because of the subjectivity it sometimes may associate. But from theoretical perspective – checking if some music obeys the rules of theory or not is quite simple and very objective task, provided that we build it on established rules. We hope that by showing another point of view we have moved the thinking a little step further and, perhaps, induce some new ideas in someone else's mind.

# Bibliography


[1] Bergstrom, T., Karahalios, K., and Hart, J. C. Isochords: Visualizing Structure in Music. *Graphics Interface 2007* (2007).

[2] Cohn, R. Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective. *Journal of Music Theory 42/2* (1998).

[3] De Haas, W. B., Magalhães, J. P., and Wiering, F. Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. *ISMIR 2012* (2012).

[4] Fujishima, T. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. *International Computer Music Conference 1999* (1999).

[5] Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*, second ed. Addison-Wesley, Boston, 2001.

[6] Johansson, K. The Harmonic Language of The Beatles. *STM-Online 2* (1999).

[7] Knuth, D. E. The Complexity of Songs. *Communications of the ACM 28/3* (1984).

[8] Laborecký, J. *Music Terminological Dictionary*. SPN, Bratislava, 2000.

[9] Lerdahl, F. *Tonal Pitch Space*. Oxford University Press, Oxford, 2001.

[10] Lerdahl, F., and Krumhansl, C. L. Modeling Tonal Tension. *Music Perception: An Interdisciplinary Journal 24/4* (2007).

[11] Lewin, D. A Formal Theory of Generalized Tonal Functions. *Journal of Music Theory 26/1* (1982).

[12] LEWIN, D. *Generalized Musical Intervals and Transformations*. Yale University Press, New Haven, 1987.

[13] MARŠÍK, L. *Ear training application*. Bachelor's thesis, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, 2010.

[14] MAUCH, M., AND DIXON, S. Approximate Note Transcription for the Improved Identification of Difficult Chords. *ISMIR 2010* (2010).

[15] MÓŽI, A. *Study Materials on Music History*. Academy of Performing Arts, Bratislava, 1994.

[16] PARDO, B., AND BIRMINGHAM, W. P. The Chordal Analysis of Tonal Music. *Computer Music Journal 26/2* (2002).

[17] POSPÍŠIL, J. *Music Theory for Music Conservatories*, vol. 1. SPN, Bratislava, 1985.

[18] RIEMANN, H. *Dictionary of Music*. Augener Ltd., London, 1896.

[19] RIEMANN, H. *Harmony Simplified*. Augener Ltd., London, 1896.

[20] ROCHER, T., ROBINE, M., HANNA, P., AND DESAINTE-CATHERINE, M. A Survey of Chord Distances With Comparison For Chord Analysis. *ICMC 2010* (2010).

[21] SCHÖNBERG, A. *Theory of Harmony*. University of California Press, Los Angeles, 1922.

[22] STARK, A. M., DAVIS, M. E. P., AND PLUMBLEY, M. D. Real-Time Beat-Synchronous Analysis of Musical Audio. *DAFx 2009* (2009).

[23] VOLEK, J. *The Structure and Figures of Music*. Panton, Prague, 1988.

[24] ZANETTE, D. H. Music, Complexity, Information. *The Computing Research Repository abs/0807.0565* (2008).

[25] Zika, P., and Kořínek, M. *Tonal Harmony for 1st-3rd Class of Music Conservatory*. SPN, Bratislava, 1990.