California Institute of the Arts

The Algorithmic Beauty of Music: Approaches to Computer-Aided Algorithmic Composition

by

Bruce Anthony Dawson

A thesis submitted in partial fulfillment for
the degree of Master of Fine Arts

in the
Herb Alpert School of Music
Music Technology: Interaction, Intelligence & Design

Supervisory Committee

Dr. Jordan Hochenbaum
Mentor

Dr. Ajay Kapur
Committee

Dr. Owen Vallis
Committee

Abstract

Since the creation of the computer, compositional experiments utilizing algorithms have pushed the boundaries of technology and musical output – fulfilling the curiosity of composers and computer scientists alike. This thesis presents several approaches to Computer-Aided Algorithmic Composition (CAAC). As such, the scope of this research thesis is to explore CAAC techniques for the future of algorithmic composition in advancing digital environments.

This research applies algorithmic processes within the domain of musical composition. In doing so, this work is concerned with uncovering interesting new forms of musical interaction between composers, computers, and algorithms through the means of Computer-Aided Algorithmic Composition (CAAC) processes. Are there specific compositional areas where algorithmic processes can benefit musical composition? In similar ways, can computer-aided algorithmic interaction support new forms of creative compositional processes?

The involvement of Music Information Retrieval (MIR) during this research has led to interesting musical results in previously underexplored musical territory with room for future expansion. Rather than using MIR features for analysis exclusively, MIR features are applied in the context of musical composition. In doing so, this research presents work utilizing MIR audio analysis techniques for the purpose of automated musical content generation.

This research also presents a variety of CAAC scenarios for musical content generation. This includes automatic melody generation using the Online Encyclopedia of Integer Sequences (OEIS) database for analysis and musical transposition to demonstrate the availability and significance of CAAC resources, such as the OEIS, in a digital compositional environment.

Furthermore, this research also presents the use of CAAC techniques in the context of Lindenmayer Systems (L-Systems). A practicing musician plays a MIDI instrument and transposes output MIDI data through the L-System software, iterating through axiom or seed mutations within the program, realizing a unique musical structure for their composition. Additional CAAC techniques are discussed in the thesis, and demonstrate new types of performance statistics obtainable from CAAC processes.

The primary contributions of this work include (1) three new software tools enabling musicians and composers to generate CAAC-based automated musical content. (2) investigating existing CAAC techniques for automatic musical content generation in digital compositional environments, (3) music

information retrieval approaches to CAAC-based automatic musical content generation; (4) lastly this research investigates the capabilities and design considerations of CAAC processes both in the digital domain, as well as the applicability of CAAC techniques in other musical domains. This work provides a overview of applied CAAC processes, as well as new CAAC techniques which can occur in the future, benefitting from the unique domain of CAAC techniques.

Acknowledgments

I would like to acknowledge and thank the universe, my family Mina, Bruce, Nicholas, Laura and Alijah; none of this would have been possible without the love patience and support of my family. Additionally, I would like to acknowledge and thank my mentors Jordan Hochenbaum, Owen Vallis, and Ajay Kapur, my close friends, my tribe, and my CalArts colleagues for the constant inspiration. I would like to thank those whom have supported my musical endeavors and educational endeavors, those that have shined their influence on me and contributed to my personal growth, directly or indirectly. Additionally, I am especially grateful to those whom have ever taken the time to believe in my goals, in my passion, and in my path. May this contribution to the academic field benefit the knowledge of our planet. May this research be inspiration for another. May all beings be happy.

Contents

Acknowledgments **6**

Chapter 1 Introduction **15**

1.1 A Definition of Terms **16**

1.1.1 A Definition of "Algorithm" 16

1.2 A Definition of "Computer Aided Algorithmic Composition" **17**

1.3 Overview of Thesis **18**

Chapter 2 Background **19**

2.1 Development of Symbols, Numeral Systems, and Zero **19**

2.2 Historical Approaches to Algorithmic Composition **20**

2.2.1 Guido d' Arezzo's Method 20

2.2.2 Tala Rhythm Systems 22

2.2.3 Athanasius Kircher's "Musurgia Universalis" 22

2.2.4 Johann Philipp Kirnberger's "Der allezeit fertige Menuetten- und Polonaiseconponist" 23

2.2.5 Approaches to CAAC 23

2.2.6 Experiments of Hiller and Isaacson 24

2.2.7 Xenakis's Stochastic Music Programme 24

2.2.8 Koenig's Project 1 (PR1) and Project 2 (PR2) 26

2.2.9 Ariza's athenaCL 28

2.3 Summary **29**

Chapter 3 MIRKov **31**

3.1 Motivation **31**

3.2 Background **32**

3.2.1 Selected Examples of Music Information Retrieval Systems 33

3.2.2 Markov Model 34

3.3 Application **35**

3.3.1 Design Overview 36

3.3.2 System Overview 36

3.3.3 Workflow 37
3.4 Example Use Case of MIRKov 40
3.5 Summary 41
Chapter 4 OEIS-TO-MIDI 42
4.1 Motivation 42
4.2 Background 43
4.2.1 Historical Example of Integer Sequences 43
4.2.2 The Fibonacci Spiral and the Golden Ratio in Popular Media 44
4.3 Application 45
4.3.1 Design Overview 45
4.3.2 System Overview 46
4.3.3 Workflow 47
4.4 Example Use Case of OEIS-to-MIDI 47
4.5 Summary 48
Chapter 5 LSysGenerator: Lindenmayer-System Chord Generator 49
5.1 Motivation 49
5.2 Background 50
5.3 Lindenmayer Systems 51
5.4 Categories of Lindenmayer Systems 51
5.4.1 Context-free (OL) Systems 51
5.4.2 Context-Sensitive (IL) Systems 52
5.4.3 Bracketed L-Systems 53
5.4.4 Propagative (PL) Systems 53
5.4.5 Non-Propagative L-Systems 53
5.4.6 Table-based (TL) Systems 54
5.4.7 Parametric L-Systems 54
5.4.8 Deterministic (DL) Systems 54
5.5 Application 55
5.5.1 Design Overview 55
5.5.2 System Overview 55
5.5.3 Workflow 56
5.6 Example Use Case of LSysGenerator 56
5.7 Summary 57

Chapter 6 Conclusion **59**

6.1 Summary **59**

6.2 Primary Contributions **60**

6.3 On Algorithmic Integrity **60**

6.4 Final Thoughts and the Future **61**

Bibliography **62**

List of Tables

Table 1.1: Five Characteristics of an Algorithm 16
Table 1.2: Ariza's Seven Descriptors of CAAC Systems 18
Table 2.1: Steps of Xenakis's SMP Musical Structure 25
Table 2.2: Parameters of Xenakis's SMP 26
Table 2.3: Parameters of Koenig's PR2 27

List of Figures

Figure 1.1: Three Components of Computer Aided Algorithmic Composition 15
Figure 2.1: Guidonian Hand 21
Figure 2.2: An Example of Guido's Method 21
Figure 2.3: "Monte Carlo" Technique 24
Figure 2.4: Illustration of Koenig's List-Table-Ensemble Principle 27
Figure 2.5: Algorithmic Sequence MIDI Output via athenaCL; image hosted by Flexatone.org 29
Figure 3.1: First-Order Markov Chain 35
Figure 3.2: MIRKov Flowchart and Communications Diagram 37
Figure 3.3: Mel Power Spectrogram 38
Figure 3.4: Harmonic Spectrogram, Percussive Spectrogram 38
Figure 3.5: Tempo Detection 39
Figure 3.6: Onset Detection 39
Figure 3.7: Chromagram 40
Figure 3.8: MIRkov MIDI Output 40
Figure 4.1: OEIS-to-MIDI Flowchart and Communications Diagram 46
Figure 4.2: OEIS-to-MIDI Use-Case 47
Figure 5.1: Context-free L-System 52
Figure 5.2: Context-Sensitive L-System 52
Figure 5.3: Bracketed L-System 53
Figure 5.4: Table-based L-System 54
Figure 5.5: LSysGenerator Flowchart and Communication Diagram 56
Figure 5.6: LSysGenerator MIDI Output 57
Figure 5.7: LSysGenerator GUI 57

Chapter 1
Introduction

"With the aid of electronic computers, the composer becomes a sort of pilot: he presses buttons, introduces coordinates, and supervises the control of a cosmic vessel sailing in the space of sound, across sonic constellations and galaxies that he could formerly glimpse only in a distant dream."
– Xenakis, I. (1992)

Musical discovery encapsulates mystery and intrigue, especially within the domain of electronic music. With the turn of a knob or the shift of electrical current, vast musical discoveries of rhythm and timbres that adhere to no traditional structure or form are awaiting the composer. Musical discovery feeds the composer's imagination in the same sense that curiosity fuels musical discovery. Adhering to the idealism of musical discovery, composer and architect Iannis Xenakis envisioned a new music compositional universe with the aid of modern electronic computers. This world, one that allows for the exploration of sounds and musical structures that once existed solely beyond the realm of twelve-tone music, natural timbres, or any other traditional music theory approaches and beyond the realm of traditional instruments, is now more than ever, a tangible compositional possibility through the use of CAAC-based compositional tools.

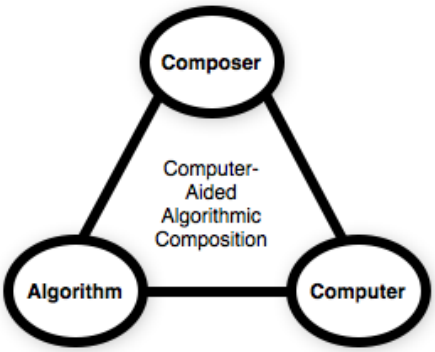


Figure 1.1: Three Components of Computer Aided Algorithmic Composition

This thesis presents several approaches to Computer-Aided Algorithmic Composition (CAAC), a synergistic field involving the interaction between composers, computers, and algorithmic processes, tied together by the means of musical expression. As such, the scope of this thesis is to explore CAAC techniques for the future of algorithmic composition in advancing digital environments.

1.1 A Definition of Terms

The concept of Computer Aided Algorithmic Composition, is derived from two terms, "computer-aided composition," and "algorithmic composition," both of which do not accurately describe the purpose and capabilities of CAAC. Music composed using a Digital Audio Workstation (DAW), could be considered computer-aided composition, however, this does not account for algorithmic processes as an integral source of the musical output. Conversely, algorithmic composition does not imply that a computer is used in the composition process – rather it implies an underlying algorithmic system that produces the musical content, with or without the aid of a computer.

1.1.1 A Definition of "Algorithm"

An *Algorithm* is defined as a methodology for covering any set of rules or sequence of operation for completing a task or solving a problem (Loy 2009). Adhering with the definition of an Algorithm as set forth by this thesis, an algorithm must demonstrate each of the following characteristics (Donald Knuth 1973):

Finiteness	The method must not take forever.
Definiteness	Each step must have a significance that is commonly understood.
Input	The method must receive valid material or information to operate accordingly.
Output	The method must produce at least one valid result, generated by the input.
Effectiveness	The method must always produce the same output from the same input.

Table 1.1: Five Characteristics of an Algorithm

Thus, for a system to be considered algorithmic, this systems methodology must compute input and output according to a set of rules, either sequentially or concurrently, and consistently in a finite, timely manner. Particularly, the definition of algorithm is essential to this research, and algorithms combined with computer-aided human interaction create the possibility for musical composition and discovery using CAAC methodologies.

1.2 A Definition of "Computer Aided Algorithmic Composition"

This research defines CAAC systems as they apply to the design of the CAAC systems detailed within, and does not attempt a comprehensive definition for CAAC methodologies. *Computer Aided Algorithmic Composition (CAAC)* is defined as a music system that facilitates the generation of new music by means other than the manipulation of direct music representation (Ariza 2005). Here, 'new music,' does not designate style or genre; rather, the output of a CAAC system must be, in some manner, a unique musical variant. Additionally specified by Ariza is that the output of the CAAC system must be that of a clearly-defined unique musical variant, such as algorithmically generated musical scores, phrases, sequences, note values, velocities, and other clearly-defined musical parameters.

CAAC systems allow the end user control over unique musical variants with the algorithmic processes involved, allowing algorithmic input to be transformed to musical output. The output of CAAC systems usually consist of incomplete musical outputs that require compositional transposition to adhere to music theory principles – conversely, some algorithmic processes focus directly on the processes of creating output rather than the musical byproduct. According to Ariza, "Such representations are indirect that they are not in the form of complete, ordered musical structures (Ariza 2005)."

Indirect by nature, several distinctions defined by Ariza are worth acknowledging that contribute to the definition of a CAAC system. Ariza organized these distinctions into seven sub-categories: (1) Scale; (2) Process; (3) Idiom-Affinity; (4) Extensibility; (5) Event Production; (6) Sound Source; (7) and User Environment. CAAC systems may adhere to the distinctions as set forth by Ariza, but are not restricted to these distinctions exclusively nor must a CAAC system adhere to all distinctions defined by Ariza. As a companion of these definitions, characteristics of these distinctions are described in Table 1.2.

Additionally adhering to the definition of CAAC systems as defined by this research, Curtis Roads' 1996 survey of "algorithmic composition systems" divides CAAC systems into four macro categories: (1) self-contained automated composition programs; (2) command languages; (3) extensions to traditional programming languages; (4) and graphical or textual environments including music programming languages (Roads 1996).

Scale	Refers to the level of musical structure the CAAC system produces.
Process	The process model of a CAAC system refers to the relationship between the computation of musical structures and their output.
Idiom-Affinity	Refers to the proximity of a system to that of a particular musical idiom, style, genre or form. Singular idiom-affinity provides tools designed for the production of music in a particular form.
Extensibility	Refers to the ability of the CAAC system to be extended. This often means adding code by form of plug-ins, modular software components, and open-source code.
Event Production	**"**A distinction can be made between the generation of events from indirect music representations (such as algorithms or lists of musical materials) and the transformation of direct music representations (such as MIDI files) with indirect models. (Ariza 2005)"
Sound Source	Refers to the ability of CAAC systems to produce event data for sound production.
User Environment	Refers to the primary form in which a CAAC system is abstracted to the user, and the framework the system is contained within allowing user configuration of these systems.

Table 1.2: Ariza's Seven Descriptors of CAAC Systems

1.3 Overview of Thesis

This thesis presents three projects that explore approaches to CAAC methodologies. These projects incorporate elements of Music Information Retrieval (MIR), integer sequence sonification, and Lindenmayer-Systems (L-Systems). Chapter 2 presents background information and historical context for this work, including the fundamental algorithms that have been explored historically, philosophies that have fueled these discoveries, and the implementation and expansion of these discoveries as a series of approaches to CAAC. Chapter 3 presents the MIRKov project, utilizing MIR features as a basis for automated musical content generation as an approach to CAAC methodologies. Chapter 4 presents the OEIS-to-MIDI project, utilizing the Online Encyclopedia of Integer Sequences (OEIS) database, containing over 250,000 algorithmic integer sequences, as a means of CAAC. Chapter 5 presents the L-System Chord Generator, utilizing the algorithmic system as a means for MIDI transposition as an approach to CAAC. Additionally, each project's chapter details motivational aspects as well as historical. This research does not attempt to account for CAAC as a comprehensive guide in either concepts or historical examples; rather, selected works and concepts are presented as a focus for exploration of CAAC systems applied.

Chapter 2
Background

"It takes a good composer to design algorithms that result
in music that captures the imagination."
– Curtis Roads, 1996

This section presents several selected approaches to Algorithmic Composition and CAAC within a historical context, leading to present day. These include numerous historical formalizations of information and the algorithmic process that has occurred and progressed over centuries being influenced by numerous disciplines and fields of studies that have helped shape the field of CAAC today. As such this chapter begins by discussing the development of number systems and symbols for mathematical articulation in section 2.1. In 2.2, historical approaches to algorithmic composition are presented, specifically focusing on the algorithmic practices themselves, rather than the musical by-product of these algorithmic practices. Finally, chapter 2.3 documents an overview of CAAC methods. While the methods discussed are not a definite comprehensive historical context of CAAC, this section documents numerous notable contributions to the field of CAAC, which appear frequently in related literature of the work presented in the rest of this thesis. At the conclusion of this chapter, several current approaches to producing CAAC-based musical content will be presented.

2.1 Development of Symbols, Numeral Systems, and Zero

In order for an algorithm to exist at the fundamental level, "the symbol must be introduced as a sign whose meaning may be determined freely, language must be put into writing and a number system must be designed (Loy 2009)." Additionally, the number zero, "as an independent number, was developed by the Indians and first handed down through an inscription from Gwalior about 400 kilometers south of Delhi (Loy 2009)." Importance is placed on the zero in an algorithmic environment, as a placeholder cannot suffice in a mathematical equation or the binary system of a computer. Additionally, the number zero is

essential to the functioning of the five defining characteristics of an algorithm as defined by this research, and is essential to CAAC approaches, in the understanding that numeral and symbolic systems are the fundamental basis of an algorithm. The number Zero has been an integral part of countless historical number systems, and additionally signifies an "off" position within binary systems, as well as signifying the first beat of a *vibhag* according to North Indian Tala Systems (Courtney 2015).

2.2 Historical Approaches to Algorithmic Composition

Music composition is a methodological process. The arts and science fields as well are algorithmic, as the processes one takes to achieve a particular outcome requires sequential and methodological action, and additionally must also adhere to definition of an algorithm as set forth by this thesis. Studying compositional methodology allows understanding of aesthetic aim within compositional practice. To understand methodology is to understand outcome. We will first discuss pre-modern computing means of algorithmic composition works in sections 2.2.1 through sections 2.2.4, followed by historical examples of approaches to CAAC, formally introducing the computer's involvement in approaches to the algorithmic compositional process.

2.2.1 Guido d' Arezzo's Method

Guido d' Arezzo made several contributions to the compositional process: he invented the solmization syllables *ut (do), re, mi, fa, sol, la;* which corresponds with the tones C, D, E , F ,G ,A, known as the hexachord (Swanson, n.d.). Arezzo created a memorization technique known as the *Guidonian hand* (Figure 2.1), and also published his composition method for students titled *Micrologus* (Loy 2009). Guido d'Arezzo's abundant contributions as an early music theorist made it possible for early composers to keep record of their work in manuscript, rather than remaining an oral tradition (Swanson, n.d.). Guido's methods are some of the earliest recorded methods of the algorithmic compositional process.

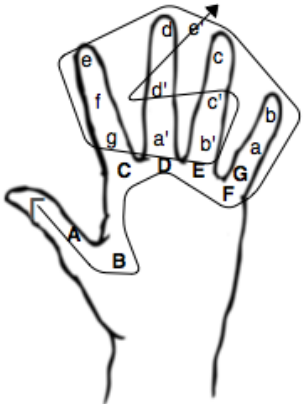


Figure 2.1: Guidonian Hand

The Guidonian method (Figure 2.2), thoroughly described in Guido d' Arezzo's 1024 manuscript known as *Micrologus*, corresponds notes of a two octaves, with the double-octave being the standard vocal scale during Guido's existence, mapping the vowel sequence, *a e i o u,* contained in Latin text to musical content generation, allowing a methodological or algorithmic transposition of text to music – the first recorded means of algorithmic composition. Following this algorithm, Guido would abstract a vowel from the analyzed text and apply the corresponding pitch to each word forming a melody. For a one-vowel text, there are three possible one-note melodies, for two-vowel text 3^2 melodies are possible, and for N-vowels, 3 melodies to the *Nth* power are possible (Loy 2009).

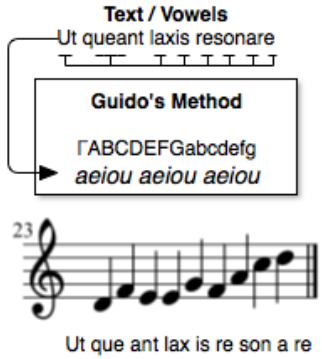


Figure 2.2: An Example of Guido's Method

According to the definition of an algorithm in the context of this research, Guido's method does not meet the criteria to be defined as an algorithm, since subjective choices are to be made using this system, and as such, it fails to meet the definiteness criteria of Knuth's definition of an algorithm. Still it is

a notable historical compositional tool, and it is to be noted that the involvement and decision-making aspect of this system, if controlled by a computer, would qualify this approach to an existing CAAC methodology applied under new mappings.

2.2.2 Tala Rhythm Systems

In Sanskrit, *Raga* literally translates to "color," or "mood," and is the melodic framework that Indian music operates within (Nettle 2015). *Tala* translates to "clap," in Sanskrit – the main rhythm or pulse of a Tala-based composition. Tala consists of beat cycles ranging from 3 beats to 128 beats per cycle (Nettle 2015), recurring with identical pattern throughout a musical performance of a particular Tala. Tala practices originate from India, however implementation of Tala systems vary from North India to South India, with South Indian practices including units of one, two, and three to seven beats, while North Indian practices allow the *Tala* to appear in units of two, three, or four beats – including both, "strong," beats and, "empty," beats (Nettl 2015). Hundreds of Tala cycles exist with limitless possibility for interchangeable rhythms. Tala cycles commonly used include *tintal* (cycle: four beat, four beat, four beat, four beat)*, rupaktal* (cycle: three beat, two beat, two beat), and *jhaptal* (cycle: two beat, three beat, two beat, three beat) (Nettle 2015). Tala systems are an example of algorithmic composition, allowing intricate interlocking rhythms applying music theory across multiple musicians and instruments. With the involvement of computers, this system could be applied as a novel CAAC technique that accounts for the applicability of Tala-based approach to CAAC systems in a digital compositional environment.

2.2.3 Athanasius Kircher's "Musurgia Universalis"

Written in 1650, "Musurgia Universalis," is a comprehensive work of musicology by Athansius Kircher, which was hugely influential in the development of western music and particularly J.S. Bach and Beethoven (Devlin 2002). The work introduces multiple methods for automated generation, one of which proposes a system consisting of three categories of labeled wooden sticks known as, *syntagmas*. These contained engravings of numbers and rhythmic values allowing for the generation of contrapuntal compositions which could be created using the systems proposed by Kircher in Musurgia Universalis (Nierhaus 2009).

Kircher's "Arca Musarithmica," proposes as system where four-lined number columns can be combined with four-voice rhythmic patterns by means of the syntagmas device which serve to transfer text passages to create four-voice movements (Nierhaus 2009). Conversely, in the Arca Musurgia, another automated generation method proposed by Kircher, the tone pitches and rhythmic patterns advance in a way that allows multiple styles of generation: church style, madrigal, motet, fugue, and monody (Nierhaus

2009). The involvement of computers could transpose this algorithmic system to a novel CAAC system, as a qualified, detailed approach to CAAC methodologies.

2.2.4 Johann Philipp Kirnberger's "Der allezeit fertige Menuetten- und Polonaiseconponist"

Designed in 1757, Johann Philipp Kirnberger's "Der allezeit fertige Menuetten- und Polonaiseconponist," was among the first record algorithmic systems, being that of the first recorded musical dice system (Nierhaus 2009). During the 18th century, dice games were a popular means of generating musical content based on the basis of combinatorial possibilities created by the dice involved. Similarly to Guido's *Micrologus* system, the composer must allocate parameters for generating musical content using the dice system – "a number of musical constellations (mostly bars) must be available out of which one can be chosen deliberately without running the danger of producing musical clashes by doing so. For example for bar 1, a number of musical arrangements of the tonic are possible. For bar 2 arrangements of the subdominant are used and for every further bar again a number of options are available (Nierhaus 2009)." Adhering to the definition of an algorithm contained within this thesis, this system, much like *Micrologus*, fails to meet the criteria to be classified as an algorithm as human decision is mandatory for musical output, however these early systems are early reflections of CAAC-based approaches to musical content generation.

2.2.5 Approaches to CAAC

Within a decade of the earliest electronic digital computer's, such as the Colossus computer (Copeland 2004), the earliest documented use of CAAC was programmed by Geoff Hill around 1955, playing simple melodies through the integrated computer speaker (Doornbusch 2004). Additionally, on July 15, 1956, the television show "Adventure Tomorrow" featured a piece of music composed by Dougles Bolitho, Martin L Klein, and Jack Owen, written with the assistance of a Datatron computer (Ariza 2005). The program required a user to enter a random ten digit number, which then would generate 1000 digits based off the input, each representing one of ten possible pitch values – pitches were then randomly selected, filtered, and output (Ariza 2005). According to this thesis' definition of an algorithm, the above compositional method fits all five traits of an algorithm and can be considered one of the first legitimate approaches to CAAC. Additionally, the following section contains compositions and experiments that qualify as approaches to CAAC.

2.2.6 Experiments of Hiller and Isaacson

Initially the experiments of Lajren Hiller and Leonard Isaacson were that of programming a computer to undertake counterpoint composition (Buxton 1977). The result of Hiller and Isaacson's experiments, which lead to the creation of *ILLIAC Suite for String Quartet* in 1957, allowed computer-aided composition involving the generation of simple diatonic melodies in the form of two and four-part polyphony (Buxton 1977). This program represented, "nearly a year of a programming and experimentation. (Ariza 2005)." Hiller and Isaacson went on to further experiment beyond the creation of *ILLIAC*. In a second experiment, four-part counterpoint was computer generated, as well as one of the first recorded attempts at computer aided "Markovian," music – music where the next note to be played is dictated by the Markov

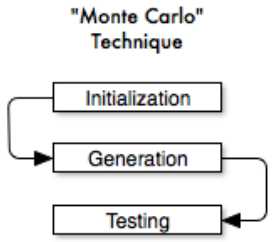


Figure 2.3: "Monte Carlo" Technique

model (further explained in Chapter 3) – where as the musical content generation is dependent on the probability parameters dictated by the Markov Model on the last notes selected. Additionally, Hiller and Isaacson's experiments used what is known as the "Monte Carlo," technique (Figure 2.3) which can be described in three steps: initialization, generation, testing (Buxton 1977).

Once rules have been created for which note combinations are considered acceptable (initialization phase), a note is generated at random (generation phase), the note is then tested (testing phase) against the rules that were "initialized," and if considered acceptable within the predefined confines of this system, then the note is appended to the score being generated (Buxton 1977). Using the above techniques and adhering to the definition of an algorithm set forth by this thesis, Hiller and Isaacson's approaches to musical generation qualifies as an approach to CAAC methodologies.

2.2.7 Xenakis's Stochastic Music Programme

Iannis Xenakis is an artist known for his pioneering of algorithmic composition and computer aided composition; Xenakis believed, "[that] With the aid of electronic computers, the composer becomes a sort of pilot: he presses buttons, introduces coordinates, and supervises the control of a cosmic vessel sailing in

the space of sound, across sonic constellations and galaxies that he could formerly glimpse only in a distant dream (Xenakis 1992)."

Stochastic Music Programme (SMP), created in 1965, lead to compositions developed based on formulae developed for simulating the behavior of gas particles, known as Maxwell's and Boltzmann's Kinetic Theory of Gases (Edwards 2011). Xenakis saw his algorithmic compositions similar to what he studied as gaseous clouds of sound, which individual notes represented the gas particles (Edwards 2011). Xenakis' compositional approach was that of a scientific approach; his compositions focus on the means of composition, being that of algorithmic distributions, means, and ranges (Ariza 2005), with musical content being the by-product. This is contrary to the traditional compositional method that finds melodic and harmonic structure to be the focus, and the arrangement being the by-product – rather than the means of content generation being the focus, such as the case in approaches to CAAC. The compositional goals of each composer that chooses to engage in CAAC methodologies may vary from composer-to-composer; with the algorithmic process being at least a partial focus for those engaged.

Step	Xenakis's SMP Instructions
#1	*Duration of sequences are calculated based on mean duration provided by user.*
#2	*The density of sounds in each sequence is defined in terms of events per second.*
#3	*Instruments for each sequence are chosen from a table of instruments organized by timbre.*
#4	*Within each sequence, start times are calculated based on the density value determined in step #2 and the previous event's start time.*
#5	*Using weighted probabilities, each event is assigned an instrument number from the ensembles calculated in step 3.*
#6	*Each event is given a pitch value within a defined range according to the pitch range of the instrument selected, and the instrument's previous note played.*
#7	*For instrument classes that allow glissando (or glide), a glissando speed is calculated.*
#8	*Each event's duration is based on a maximum, the sequence "density," and the probabilities used in the selection of the timbre class, and instrument class.*
#9	*Each event is assigned a performance dynamic selected from forty-four possibilities.*

Table 2.1: Steps of Xenakis's SMP Musical Structure

Parameter	Xenakis's SMP Parameters
#1	*Start Time*
#2	*Timbre class*
#3	*Instrument number*
#4	*Pitch*
#5	*Glissando values*
#6	*Glissando values*
#7	*Glissando values*
#8	*Duration*
#9	*Dynamic Value*

Table 2.2: Parameters of Xenakis's SMP

Xenakis's SMP musical structure is based on a series of nine steps, and the SMP divides a musical event into nine parameters, demonstrated in Table 2.1 and Table 2.2 (Ariza 2010). This CAAC system is procedural and as well adheres to a particular style of musical composition (refered to as 'Idiom-Affinity," in adherence with Azria's seven descriptors of a CAAC system).

2.2.8 Koenig's Project 1 (PR1) and Project 2 (PR2)

The use of the computer for compositional means, "is a logical result of historical developments (Koenig 3)." In Koenig's PR1, users input six tempo values, twenty-eight rhythmic values, a random generator seed value, and the desired length of the composition (Hollerweger 2015). Koenig's PR1 produced a number of variants depending on the given input (Koenig 1971). According to Koenig, not much influence existed on the rules and individual data of the program, and Koenig considered that the project may not be of much interest to other composers due to the limited control of the system (Koenig 1971). As an early approach to CAAC methodologies this historical example does not entirely adhere to the definition of CAAC as set forth by this research, however PR1 is noteworthy as the predecessor of Koenig's PR2, a more robust, modern approach to CAAC techniques.

Parameter	Function
#1	Instrument
#2	Harmony
#3	Register
#4	Entry Delay
#5	Duration
#6	Rest
#7	Dynamics

Table 2.3: Parameters of Koenig's PR2

Koenig's PR2 was the next development, involving seven user-affected parameters (see Table 2.3, and the user can provide as much data as see fit for each parameter and the, "several sub-programmes select the data according to various principles and assemble them to form the score (Koenig 1971)."

PR2's guiding principles were defined as such: (1) to be able to describe the infinite possibilities of computers through the language of music; (2) to enable composers a platform to write music that is less error-prone; (3) and eliminating the need for the user to be technically-savvy with programming, algorithmic composition, or mathematics.

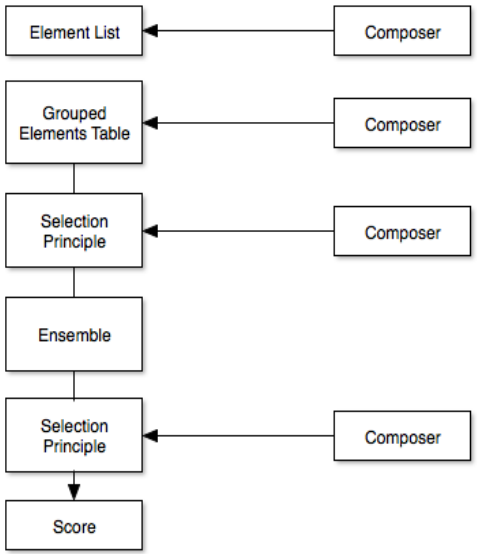


Figure 2.4: Illustration of Koenig's List-Table-Ensemble Principle

Koenig's "List-Table-Ensemble principle (Koenig 1971)," was developed and applied to five of the seven parameters (Table 2.3), registering the elements of the parameters from the list of parameters to be combined into groups, which then the composer can preliminarily sort through. As Koenig's PR2 program contains multiple accessible parameters allowing thorough control and manipulation of the system, a CAAC approach is formed: allowing combinations of human-interaction, computer-interaction, and algorithm-interaction to synergistically create automated musical content not previously possible by means provided in the PR1 project.

2.2.9 Ariza's athenaCL

Christopher Ariza's "athenaCL" is a command-line CAAC program written in the Python programming language with the ability to output musical content either as general MIDI format, or as audio via the Csound programming language. The athenaCL system features specialized objects for creating and manipulating pitch structures, including the Pitch, the Multiset (a collection of Pitches), and the Path (a collection of Multisets) objects (Ariza 2010).

This comprehensive CAAC tool combines any number of user-defined textures, allowing composition with provided tools for generative grammar systems, Lindenmayer systems (L-systems), fractional noise, cellular automata, and other CAAC models – athenaCL can be considered a meta package of CAAC tools, presenting numerous approaches to CAAC techniques.

According to the athenaCL tutorial manual: within the athenaCL command-line environment, musical parts are deployed as "PathInstances," and "TextureInstances," PathInstances are objects within athenaCL that define pitch materials, and TextureInstances are defined by athenaCL as objects containing "ParameterObjects," which control musical parameters such as amplitude, panning, rhythm, and tempo (Ariza 2010). The athenaCL environment is compatible with multiple digital formats including: Csound, SuperCollider, MIDI, and additional formats. Examples of algorithmic sequences output by athenaCL are demonstrated in **Error! Reference source not found.**

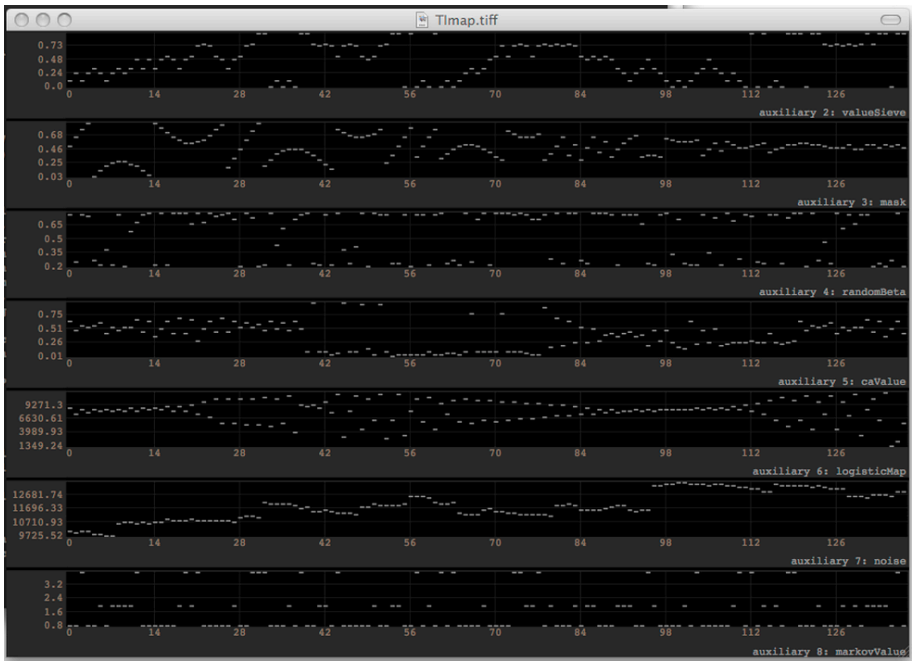


Figure 2.5: Algorithmic Sequence MIDI Output via athenaCL; image hosted by Flexatone.org

2.3 Summary

Algorithmic composition techniques have been of interest to musicians for hundreds of years – with every iteration of a new algorithmic composition system seemingly building upon the foundation knowledge of algorithmic composition systems applied historically. Not until the past hundred years, has the involvement of the computer in algorithmic compositional systems been a reality; with the involvement of the computer, the CAAC-driven approach is not only one of composition, but that of musical discovery and manipulation within the context of the given CAAC systems. The field of CAAC has made continuous effort to adapt new methodologies for creating CAAC-based musical content, encouraging more effective communications between humans, computers, and algorithms within the domain of CAAC. Musical composition usually implements a rule-based algorithmic system to adhere to music theory guidelines, such as twelve-tone music theory, for example: a time signature and key might be declared for a musical composition – an algorithmic approach ensuring musical content creation adheres to the guidelines of the designated key, and have an even metering, or timing throughout the entire composition; if either key or meter is to change within the system, it is declared in a symbolic, algorithmic, clearly defined manner consistent with the parameters set forth by the applied system. Thus, this chapter has provided examples of historically recorded approaches to CAAC techniques that adhere to music theory practices worldwide.

In this chapter, an overview of significant historical works within the domain of algorithmic composition and CAAC was provided. Effective CAAC methodologies require the ability for the user to input data, and the ability for the computer to understand this input, adhering to the definition of an

algorithm set forth in the context of this thesis, and generate meaningful automatic musical content that can be re-shaped and re-discovered by new user input. Within the context of this thesis it is believed that through the exchange of musical information and musical parameter manipulation as set forth by the combination of algorithmic interaction, computational interaction, and human interaction, that novel CAAC systems are possible, and are a tangible compositional process leading to unique musical results.

Chapter 3
MIRKov

"What was once only an abstract concept – like math – now has a window into it that has many components from the visual arts. What this means by extension is that computers will make our future adult population much more visually literate and artistically able than today."
– WIRED (1994)

3.1 Motivation

As the distinction between computer science and artistic expression becomes further integrated daily in the midst of advancement of multimedia expressionism through means of digital art, the hacker becomes an artist, the composer becomes an engineer. "My belief in this comes from watching computer hackers, both young and old. Their programs are like paintings: they have aesthetic qualities and are shown and discussed in terms of their meaning from many perspectives. Their programs include behavior and style that reflect their makers. These people are the forerunners of the new expressionists (WIRED 1994)."

This quote is, to-date, over twenty years old, however, it holds true to form in regard to digital art more than ever in present time. Computer processes that once took minutes-to-hours to render, can now be done in real time, and the increase in processing power opens doorways to possibilities that technology could not support years ago. One embodiment of the outcomes resulting from the increased effectiveness of computer processing has been the birth and rapid advancement in the field of Music Information Retrieval (MIR), a domain of computer science concerned with extracting meaning information for audio and sound. MIR "features" automatically extracted from a piece of audio can be low-level features, e.g. describing the spectral characteristics of a sound frequency makeup, or high-level, essentially mapping low-level features into higher level human-musical constructs. This thesis asserts that music composition through the use of computer science allows further computation of audio signals for creative purposes. The MIR field has traditionally focused analysis of audio or musical sources on to a plethora of tasks such as (but not limited to) sound/audio/music identification and classification, sound-source separation, pitch

identification, style identification or classification, automatic transcription, musical fingerprinting, etc. In this work however, MIR features are applied directly to the task of musical composition in the context of CAAC systems.

The MIRKov project aims to bridge the gap between MIR practices and composition practices, allowing users to analyze audio through MIR approaches and create musical relations within the analysis with the use of the Markov model, explained later in this chapter, allowing for melodic content generation via MIDI output. MIRKov allows the composer to combine computer science with the musical compositional process, creating a synergistic computer-aided compositional tool for musical exploration adhering to the domain of CAAC and exploring MIR features once reserved exclusively for musical analysis and research. In accord with Ariza's descriptors of CAAC systems as presented in Chapter 2, a primary goal of this research is to explore the combinations of MIR features and CAAC features as a real-time CAAC system, adhering to the principles of an algorithm as set forth by this research and allowing for new means of artistic musical expression as an approach to CAAC.

3.2 Background

Analytical methods for automated musical examination have become accessible and computable within the past century, including methods such as the *Fast Fourier Transform* (FFT), Spectrograms, and the Chromagram, having been created involving methodologies available only with the calculatory aid of a computer. MIR is described as "a multidisciplinary research endeavor that strives to develop innovative content-based searching schemes, novel interfaces, and evolving networked delivery mechanisms in an effort to make the world's vast store of music accessible to all (Downie 2004)."

In the interest of coherent presentation, and not intended as a comprehensive collection of definitions of the presented terms, previously mentioned MIR features will be defined here. FFT-based techniques are commonly used in MIR for spectral analysis of frequency in a recorded piece of audio; allowing transformation of space from the time-domain to frequency, and vice versa (Van Loan 1992). Pitch-tracking is made possible by real-time audio analysis using FFT (Brown 1990), as well as features which build further upon pitch-tracking such as pitch correction and harmony detection (Pfeiffer, Fischer, and Efelsberg 1997). Additionally, FFT Calculation has had an impact on many different fields of research and according to Charles Van Loan, "It is generally accepted that 'Life as we know it would be very different without the FFT' (Van Loan 1992)."

Expanding on the MIR technique of FFT, Spectrograms are created using FFT feature extraction and divided into analysis windows, calculating the magnitude of an analyzed audio source. Spectrograms can be defined as an intensity plot (Smith III 2011), and is extensively used within the domain of speech processing and phonetic identification (Flanagan 1972).

Musical key, in addition to tempo and genre are important musical attributes to the analytical process and with the use of a chromagram, accurate automated key detection is a possibility utilizing automated MIR features. Based on a model of human auditory perception, the chromagram is a straightforward MIR approach (Pauws 2004). Key detection is a beneficial focus of automated music analysis, and additionally is the basis for further MIR analysis.

3.2.1 Selected Examples of Music Information Retrieval Systems

Information Retrieval (IR) has been utilized by powerhouse companies such as Google, Yahoo, eBay, and the likes through the use of their expansive search engines – able to parse through some of the world's largest databases, remotely, in a matter of seconds. IR methods are useful for retrieving data from sources between text such as image, video, and music. Music Information Retrieval (MIR) is a useful subdomain of the IR field that "requires a deep understanding of theoretical, analytical, and practical approaches in its various environments [such as music theory and computer science] before researchers can learn how and why it must be accessed (Meurer 2004)."

Popular consumer methods of MIR consist of programs such as Pandora (http://www.pandora.com), Shazam (http://www.shazam.com), and tools such as Musipedia (http://www.musipedia.org). Pandora, for example, presents the consumer, or end-user, with instructions, "type in the name of your favorite artist or song and we'll create a radio station featuring that music and more like it (Casey 2008)." The Pandora system then uses MIR methods to estimate similarities between inputted artists and similarities between this artist's music and others like it. The Pandora system is an example of a working MIR system – with a database consisting human-entered track-level metadata, which then the MIR Pandora system makes comparisons between similarities in this database to make music selections for the end-user (Casey 2008). Additionally, during the metadata input process, entering each track into the database can take up to 20-30 minutes of an expert's time – taking approximately 50 person-years to enter the metadata for one-million tracks (Casey 2008).

The popular smartphone application, Shazam, is known for allowing users to quickly and efficiently identify music and television with the use of a smartphone microphone and audio fingerprinting technology (Cano 2002) in conjunction with unique MIR feature extraction techniques. Shazam's

proprietary technology sends the recorded audio to a database filled with millions of tracks, attempting to identify the song and return the correct result to the user, allowing the user access to purchase the music, or download ringtones if available. With accurate results and a solid MIR algorithm, Shazam now, "connects over over 500 million people (http://www.shazam.com/company)." Shazam was created in 2000, with no existing algorithms at the time that would tackle the problem Shazam was attempting to resolve; Shazam's own technique was then created. Shazam's music identification algorithm encompasses a few MIR techniques, not limited to but including: Spectrograms, Combinatorial Hash Generation, Constellation Maps, Hash Details (Wang, n.d.).

Over the years similar services have sprung up, however Shazam's unique MIR algorithms have kept this service widely-used. Shazam's MIR algorithm is defined by Shazam's Founder and Chief Scientist, Avery Wang, as, *"...noise and distortion resistant, computationally efficient, and massively scalable, capable of quickly identifying a short segment of music captured through a cellphone microphone in the presence of foreground voices and other dominant noise, and through voice codec compression, out of a database of over a million tracks. The algorithm uses a combinatorially hashed time-frequency constellation analysis of the audio, yielding unusual properties such as transparency, in which multiple tracks mixed together may each be identified. Furthermore, for applications such as radio monitoring, search times on the order of a few milliseconds per query are attained, even on a massive music database.* (Wang 2003)"

3.2.2 Markov Model

Andrey Andreyevich Markov (1856-1922) introduced the Markov Model as a Russian mathematician studying the field of number theory, analysis, and probability theory (Nierhaus 2001). The first known application of the Markov Model takes place in a textbook published by Markov in 1913, 'Ischislenie veroyatnosetej,' where, "A. A. Markov, studied the sequence of 20,000 letters in A. S. Pushkin's poem "Eugeny Onegin", discovering that the stationary vowel probability p = 0.432, that the probability of a vowel following a vowel is p1 = 0.128, and that the probability of a vowel following a consonant is p2 = 0.663 (Basharin et al., 2004).". In the same article, Markov also gave the results of his other tests; he studied the sequence of 100,000 letters in S. T. Aksakov's novel "The Childhood of Bagrov, the Grandson". For that novel, the probabilities were p = 0.449, p1 = 0.552, and p2 = 0.365 .

The "Markov Chain," is a known method for the branch of mathematics known as *Stochastic procedures,* composed of probability calculus and statistics methodologies, first used in 1926 by another Russian mathematician Sergey Natanovich Bernstein (Nierhaus 2001). According to Nierhaus, "*Stochastic processes* are used to describe a sequence of random events dependent on the time parameter (t). The set of events is called 'state space,' while the set of parameters is known as the *'parameter space.'* If a stochastic

process consists of a countable number of states, then it may also be refereed to as a stochastic chain. In a stochastic chain, every discrete time *t* has a random variable *X*. In a Markov *chain,* it begins a special kind of stochastic chain, the probability of the future state $Xt + 1$ (the random variable x*X* at the time t +1) depends on the current state Xt. For the given times tm and tm+1, this probability is:

$$P(X_{tm+1} = J \mid X_{tm} = i) = P_{ij}(t_m, t_{m+1})$$

Equation 1: Stochastic Chain

(Nierhaus 2001)."

First-order Markov chains (Figure 3.1) are influenced by the previous state exclusively, while Markov chains that are influenced by more than only the previous state are known as *higher-order Markov chains*, which allows for predictive transitions based on *x* number of previous states. Only previous states are relevant to the transition probabilities can be produced by the Markov chain regardless of *order,* meaning that if a particular sequence does not occur in the Markov chain input, then this sequence cannot occur through transition probability – Markov chains will only produce output based on the input, representing finite number of states, which therefor must be represented by finite input.

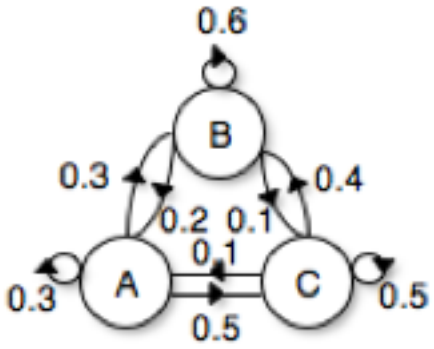


Figure 3.1: First-Order Markov Chain

3.3 Application

MIRKov aims to account for the traditional methods of MIR feature extraction as an approach to CAAC, allowing for thorough analysis of selected musical content before the computer-aided compositional process begins. As the feature extraction process can be time consuming – the first portion of a two-part process, of the MIRKov program is not designed for live performance; rather, intended as an approach to CAAC methodologies in a compositional environment. The MIRKov application is a command-line tool

utilizing the Python and ChucK (Wang and Cook 2003) programming languages which communicate via Open Sound Control (OSC) network communication protocol (Wright 1997). The greatest feat of MIRKov is the ability of the digital composer to compose within their native digital audio workstations, utilizing the musical output of MIRKov which sends MIDI data to any Digital Audio Workstation (DAW) accepting incoming MIDI signal.

3.3.1 Design Overview

MIRKov is a command-line application built to extract the musical features, such as tempo, pitch, note length, from an audio source, which is then fed into a Markov model for CAAC. The software is designed to be user-friendly, while providing support for numerous data types. As such, the following list provides an overview of the main software requirements:

- Support for a variety of audio input files (.WAV, .MP3, .AIFF, .FLAC, et cetera).
- Minimal programming required (path to audio file, and tempo command-line input only).
- MIDI Output to external hardware or virtual MIDI (e.g. Apple IAC Bus), for use with other music software and DAWs.

3.3.2 System Overview

The general flow of this software is documented in Figure 3.2. A user provides an audio source to be analyzed by the MIRKov system, and by default, MIDI is output through the specified MIDI out port as source material for a CAAC-based approach to the compositional process. MIDI Output is based on MIR analysis of the user provided file – based on tempo, pitch, notes found, and note lengths.

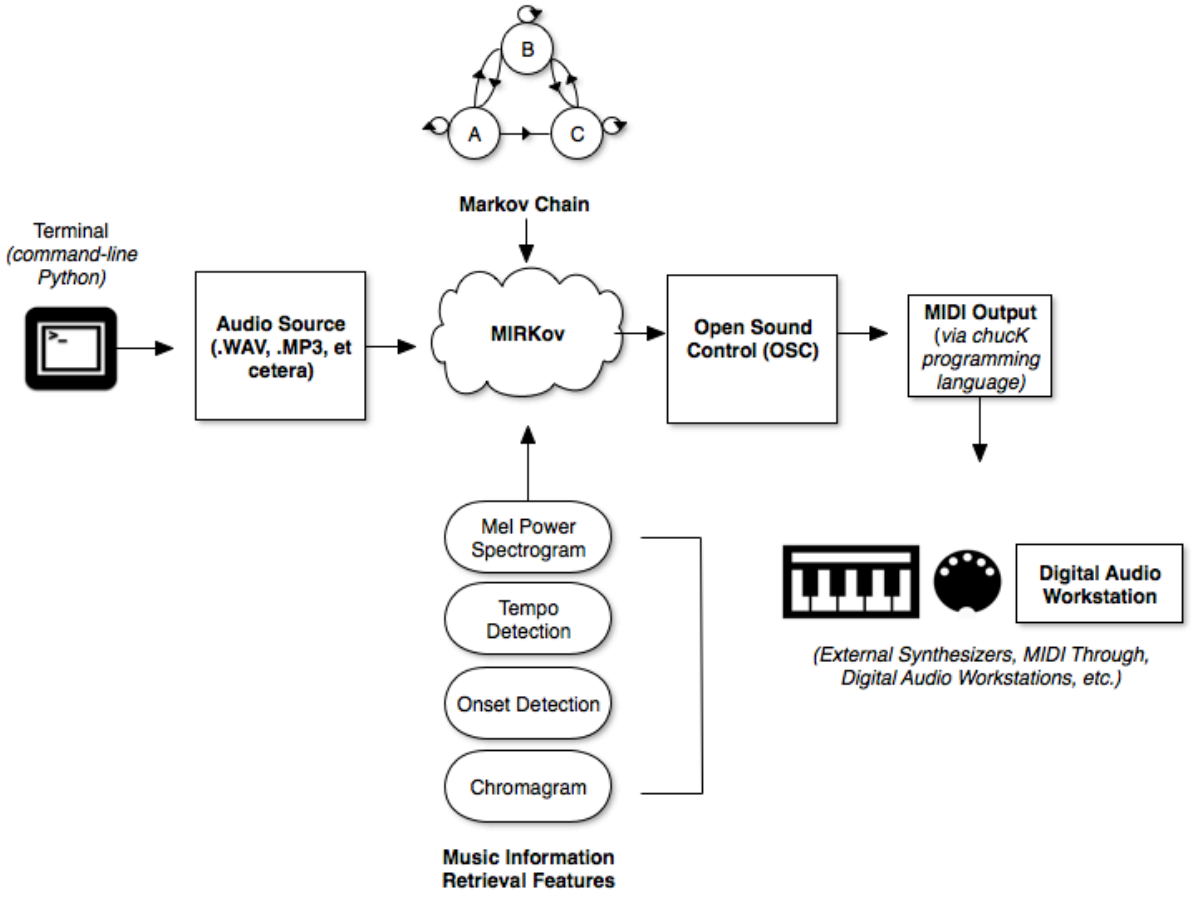


Figure 3.2: MIRKov Flowchart and Communications Diagram

3.3.3 Workflow

A typical use scenario begins with using command-line prompt to select an audio source, and enable MIRKov analysis via Terminal. Tempo can be distinguished via command-line prompt, and if the user is using Ableton Live 9 DAW with the LiveOSC plugin, tempo information can be sent via OSC from the MIRKov system directly to Ableton Live. Figure 3.2 demonstrates an example MIRKov session with the MIR feature extraction occurring without user-interaction. Optionally, multiple graphs of the MIR features extracted can be inspected prior to MIDI output including a mel power spectrogram (Figure 3.3), which can be divided into a percussive mel power spectrogram and a harmonic mel power spectrogram (Figure 3.4) using the powerful Librosa analysis library for Python. Additionally, MIRKov also provides us with Tempo detection (Figure 3.5) and onset detection (Figure 3.6), and a Chromagram (Figure 3.7) for key

detection. Lastly, MIDI is output through the designated MIDI output port, e.g. virtual IAC midi bus – compatible with any DAW that can receive MIDI.

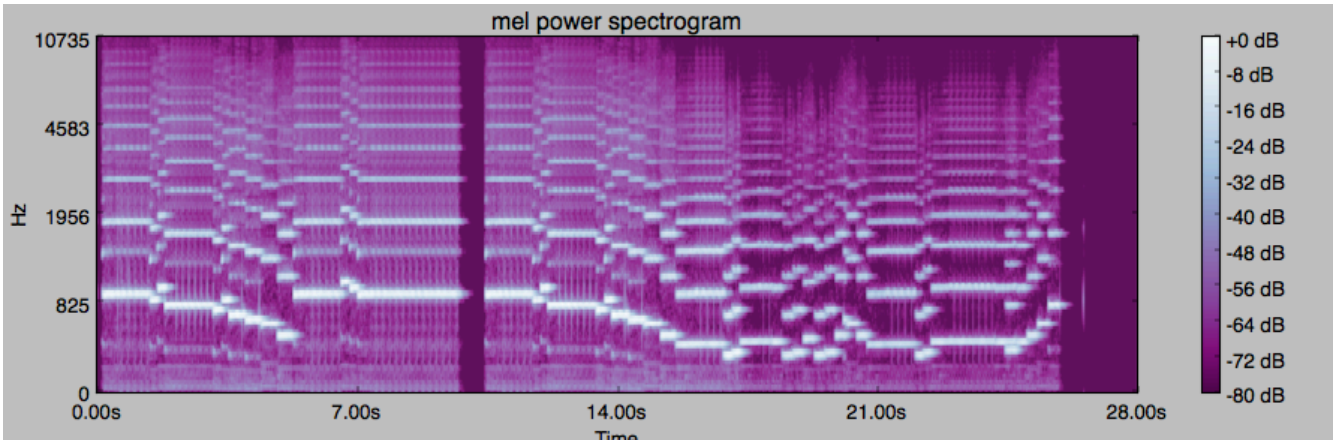


Figure 3.3: Mel Power Spectrogram

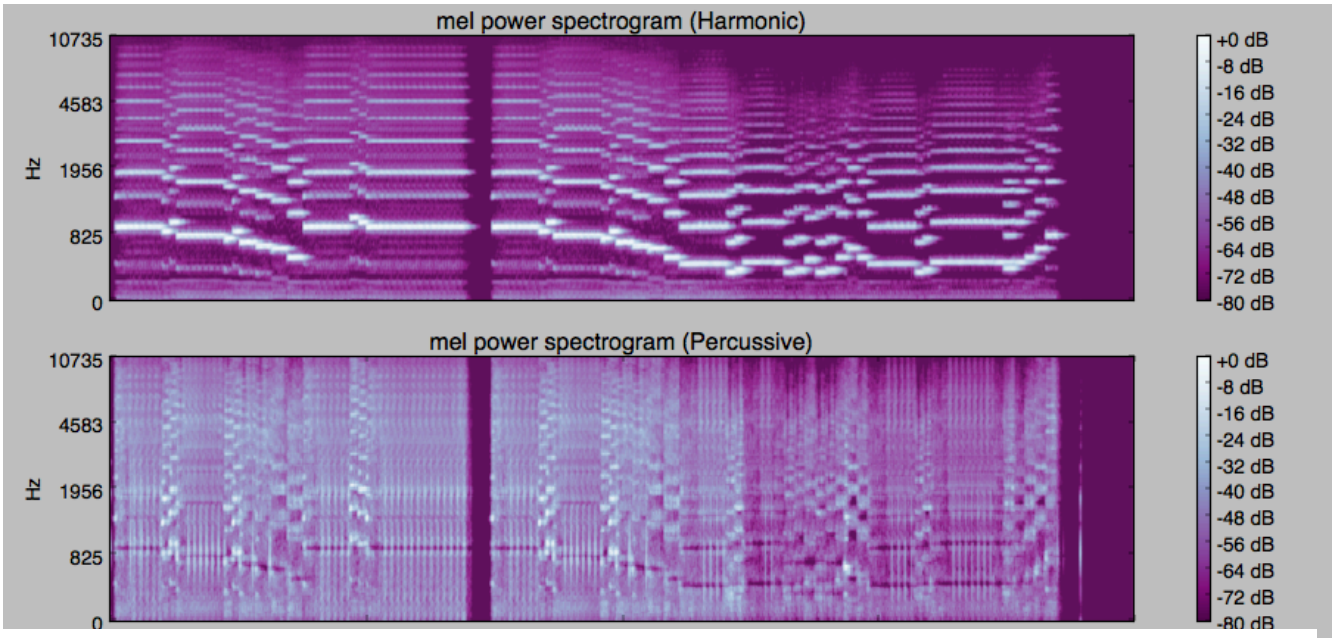


Figure 3.4: Harmonic Spectrogram, Percussive Spectrogram

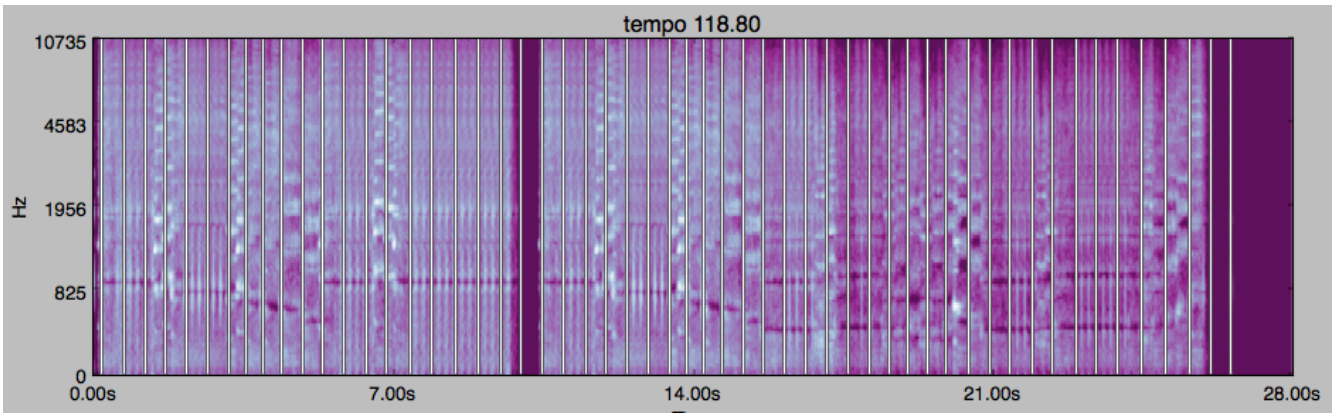


Figure 3.5: Tempo Detection

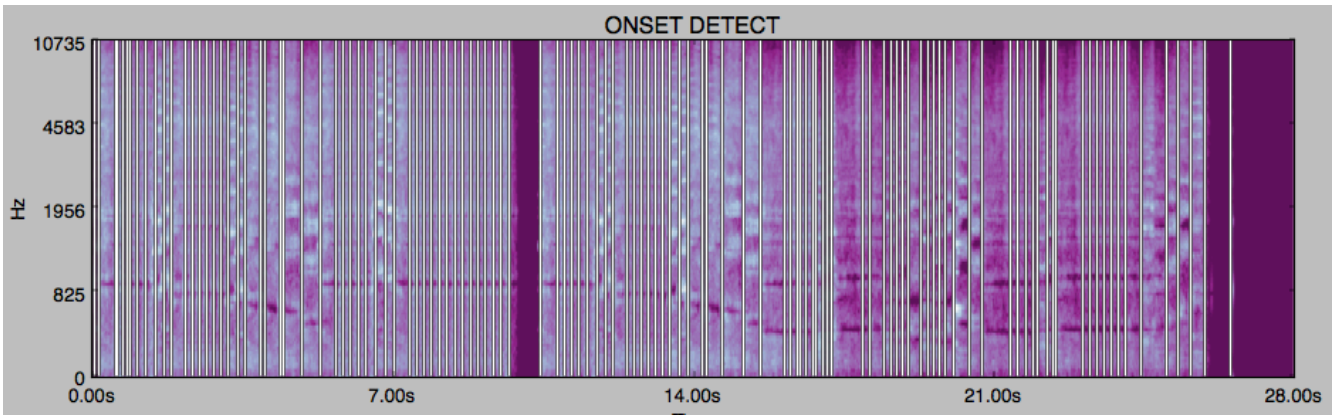


Figure 3.6: Onset Detection

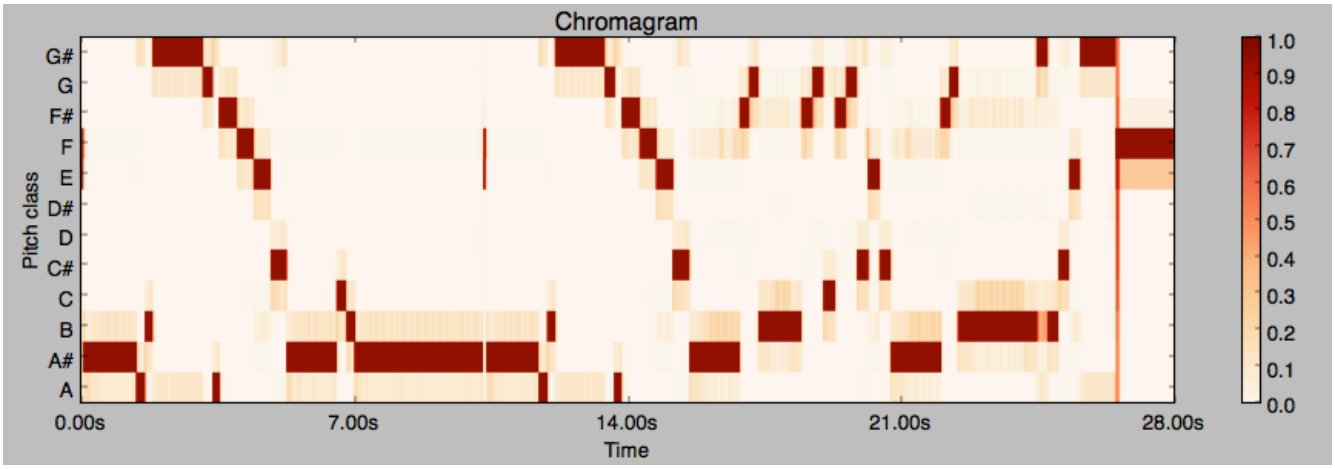


Figure 3.7: Chromagram

3.4 Example Use Case of MIRKov

An example use case of MIRKov would output MIDI notes according to the notes and note lengths detected in the analyzed audio source. MIRkov does not replicate the analyzed sound source, rather MIRkov outputs melodic content consistent with the style of the analyzed file, imitating notes and note lengths found in the source, consistent with the key or chords found in the analyzed content. Additionally, every iteration of MIRkov will output new material, allowing limitless exploration of sound sources.

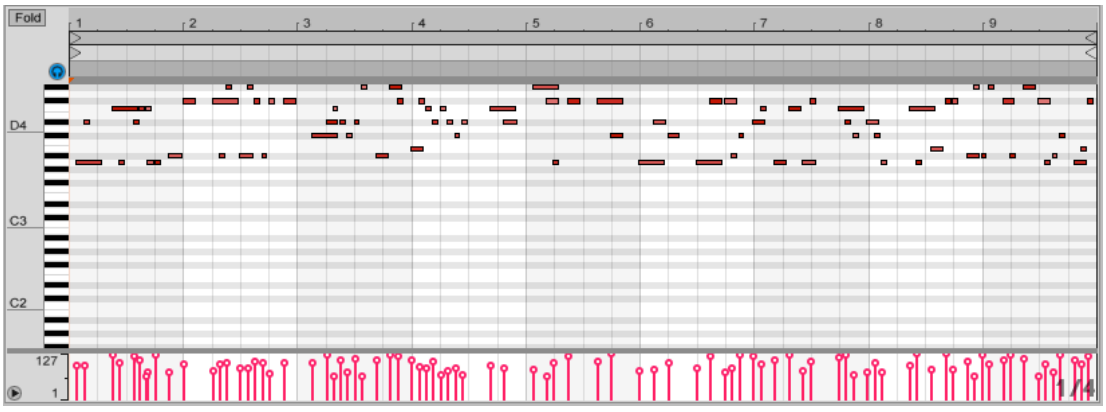


Figure 3.8: MIRkov MIDI Output

3.5 Summary

This section described MIRKov, a CAAC software tool for generating computer-aided melodic MIDI content in a digital environment based on MIR feature extraction of a specified audio source. MIRKov is different than other MIR feature extraction programs, as well as different from other melody generation programs in that it is a hybrid MIR tool for CAAC-based compositional methodologies that highlights the compositional qualities of MIR feature extraction. We look forward to a future where feature extraction is integrated into the general workflow of CAAC for digital composers. Working with feature extraction enables many unique artistic possibilities, from future applications of direct sound parameter manipulation to the current possibility and application of generating new melodies as an approach to CAAC based on the analysis of selected musical content. Not only can MIRKov increase productivity in CAAC scenarios, but we hope this project points and establishes a foundation for other feature extract based compositional tools, in the MIR field, the music studio, and the academic realm. Additionally, development of this project will continue using the C++ language to further assist system compatibility and system performance, as well as further integrating MIR feature extraction technologies into the digital compositional environment.

Chapter 4
OEIS-TO-MIDI

4.1 Motivation

Integer sequences are sets of recognizable patterns that humans begin to recognize by an early age – with sequences and patterns being found in rhythm or recognizable repetition of words, music, and numbers. Integer sequences, such as spirals angles, which have been the subject of musical stipulation over the years. Additionally the Online Encyclopedia of Integer Sequences (OEIS) has been created and updated to current date allowing a complete index of known sequences. Integer sequences are achieved through algorithmic means, thus falling into the domain of CAAC.

The purpose of the OEIS is a reference work for identifying sequences, and an archival index of known sequences and sequence algorithms. Sequences can be searched by entering the terms into the search function of the OEIS, and the OEIS will return sequences that match the terms entered, as well as the definition of the sequence, formulas for generating the sequences, references to books or scholarly references the sequences, as well as examples explaining the terms of the sequence and the name of the person whom submit the sequence to the OEIS.

Musical transposition of sequences found in the OEIS has been a compositional process in the past, with transposition of integer sequences without tools such as OEIS-to-MIDI, a CAAC system that grants instant access to every known OEIS sequence within the OEIS database for musical transposition. In accord with Ariza's seven descriptors of a CAAC system, a primary motivation and purpose behind the OEIS-to-MIDI program is having an extensible, real-time CAAC system to produce MIDI output based on the algorithmic output contained within the OEIS library. A few sequences have been popularly transposed to music, such as [A00000] which is transposed musically and is the musical backdrop to the *OEIS Movie* (OEIS Foundation Inc., 2011), featuring plots of the first 100 sequences of the OEIS. OEIS-

TO-MIDI allows quick user-friendly OEIS mappings to musical parameters – allowing seemingly limitless combinations of sequences for a lifetime of possibilities for musical exploration.

4.2 Background

While working on his dissertation as a graduate student at Cornell University in 1964, Neil J. A. Sloane (the originator of the OEIS) encountered a new sequence of numbers (now known as sequence, A000435) while searching, "for a formula for the n^{th} term, in order to determine the rate of growth of the terms. (OEIS Foundation Inc., 2011)". The sequence investigated was not mentioned by books in the Cornell library, thus the sequence, along with others, were then recorded on file cards, eventually making their way into a book in 1973 known as, "A Handbook of Integer Sequences (Sloane, Academic Press 1973)." Revised editions lead up to, "The Encyclopedia of Integer Sequences (Sloane, Simon Plouffe, Academic Press 1995), and eventually today's version known as the Online Encyclopedia of Integer Sequences (OEIS) featuring and constantly maintained by the OEIS Foundation, Inc., featuring over 250,000 integer sequences (OEIS Foundation Inc., 2011).

4.2.1 Historical Example of Integer Sequences

Intrigue into integer sequencers came well before Sloane however. Integer sequences have been a fascination of mathematicians for centuries. One of the earliest-known integer sequences was discovered by mathematician Leonard Euler (1707 – 1783) during his application of the *Pentagonal number theorem*, which is recounted in his paper "Discovery of a most extraordinary law of the numbers concerning the sum of their divisors" (Nguyen 2011). In the context of this thesis, this integer sequence is not the oldest-known historical integer sequence, and is not a comprehensive history of integer sequences, or a comprehensive recollection of mathematical formulas – rather a mathematical contribution that is relevant to the subject matter of this research. Euler's *sum of divisors* (commonly known as *Sigma*) function is an interesting mathematical object allowing terms to be computed from the foregoing terms (a property commonly found in integer sequence algorithms). The discovery of the *sum of divisors* (listed in the OEIS as sequence A001318) function involves the sequence $\{1,2,5,7,12,15,\dots\}$, which Euler observed as a sequence containing, "a mixture of two sequences with a regular law," containing odd terms $\{1,5,12,\dots,\}$ which are *pentagonal numbers* following the formula $n(3n-1)/2$, whilst the even terms $\{2,7,15,\dots\}$ follow the formula $n(3n+1)/2$. This allowed Euler to account for the n^{th} term whether it is an odd or even integer (Nguyen 2011). According to Nguyen, the following is Euler's explanation of his recognition of the divisor sequence:

"I confess that I did not hit on this discovery by mere chance, but another proposition opened the path to this beautiful property … In considering the partition of numbers, I examined, a long time ago, the expression:

$$(1-x)(1-x^2)(1-x^3)(1-x^4)(1-x^5)(1-x^6)(1-x^7)(1-x^8)$$

in which the product is assumed to be infinite. In order to see what kind of series will result, I multiplied actually a great number of factors and found

$$1-x-x^2+x^5+x^7-x^{12}-x^{15}+x^{22}+x^{26}-x^{35}-x^{40}$$

The exponents of x are the same which enter into the above formulas; also the signs + and – arise twice in succession. (Nguyen 2011)"

4.2.2 The Fibonacci Spiral and the Golden Ratio in Popular Media

Integer sequences, and particularly the well-known Fibonacci sequence $\{0,1,1,2,3,5,8,13,21,\dots\}$ has been a subject of pop culture media-manipulation for years – appearing in music, television and films, and modern architecture. The examples provided demonstrate popular references to integer sequences and do not account for approaches to CAAC, as the sequences chosen were explicitly done so with or without the aid of a computer; additionally algorithmic means did not support the output, rather the integer sequences chosen were directly mapped to musical or architectural parameters. In the context of this thesis, CAAC is an approach which requires decision making by a computer source, an algorithmic source, and a human source. The following examples of multimedia uses of the Fibonacci sequence is not intended to be a comprehensive historical context of multimedia integer sequence mappings, rather a collection of examples considered relevant to the subject matter of this thesis.

The Fibonacci sequence has been the subject matter of a number of musical experiments. While none of these examples convey parameters of a CAAC system, it is worth mentioning the influence of integer sequences has had on art, and specifically the Fibonacci sequence. In recent years, the song "Lateralus," by Tool contains the Fibonacci sequence during the first verse of the song $\{1,1,2,3,5,8,5,13,13,8,5,3\}$, and in the second verse the following sequence $\{2,1,1,2,3,5,8\}$ is also represented (DiCarlo 2001). The Fibonacci numbers are also featured in the sections of Debussy's *Image,*

Reflections in Water, whereas the key sequences are marked by the 34th , 21st, 13th, and 8th intervals (Smith 2003). Additionally, the chorus of the song *Astronomy (8th Light)* By musicians Mos Def and Talib Kweli references the third through seventh term of the Fibonacci sequence {…,1,2,3,5,8,…} in the chorus of this song:

"Now everybody hop on the one, the sound of the two,
It's the third eye vision, five side dimension,
The 8th Light, is gonna shine bright tonight"

The Fibonacci sequence has also appeared in popular cinema and literature, such as an appearance as a purposely-rearranged Fibonacci-based anagram in the literature and film, *The Da Vinci Code* written by Dan Brown in 2003, and later adopted by Sony Pictures as a film in 2006. Fibonacci also makes appearances in the 1998 film *Pi*, and the 2008 film *21*; additionally, the Fibonacci sequence and has been an Architectural centerpiece during the design of the *Eden Project*, near St Austell, Cornwall, England (Smith 2007).

While the above examples are demonstrations of integer sequences being utilized in popular culture and media and are relevant to the history of Integer sequences and their musical applicability, none of the examples above account for the criteria to be considered a CAAC-based approach.

4.3 Application

OEIS-to-MIDI aims to account for the applicability of integer sequences as a source of musical input in a CAAC-based approach to music composition. The OEIS-to-MIDI programs allows for users to use any integer sequence found within the OEIS database (which at present time contains over 250,000 integer sequences), and use this as musical material, in combination with other integer sequences for additional algorithmic user-control and musical manipulation.

4.3.1 Design Overview

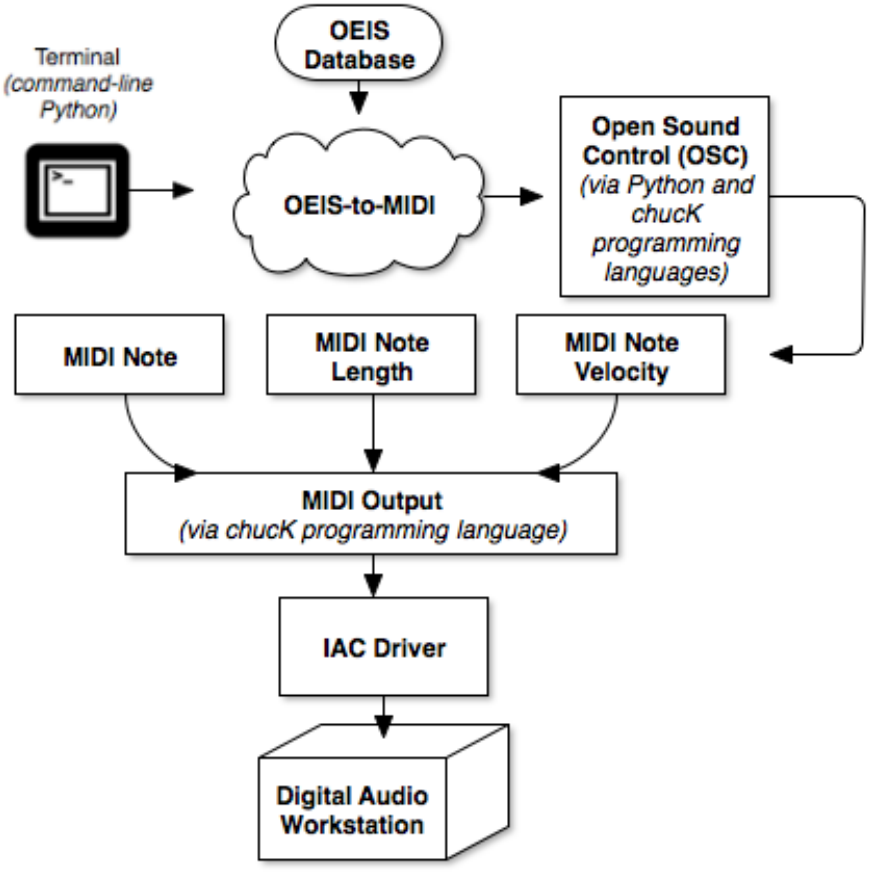
OEIS-to-MIDI is a command-line application built to extract integer sequences from the OEIS database, using these integer sequences as means for musical content generation, as an approach to CAAC methodologies. The software is designed to be user-friendly, and as such, the following list provides an overview of the main software requirements:

- Support for input of up to three integer sequences (MIDI Note, Note Velocity, Note Length).

- Minimal programming required (integer sequence ID and tempo input needed only).
- MIDI Output for use with MIDI-Compatible DAW, Synthesizers, and MIDI devices.

4.3.2 System Overview

The general flow of this software is documented in Figure 4.1. A user provides up to three integer sequence IDs and a target tempo to the OEIS-to-MIDI program, and by default, MIDI is output accordingly. Bus as source material for a CAAC-based approach to the compositional process. Integer sequences are mapped to MIDI note, MIDI note length, and MIDI velocity. If a single sequence is input only MIID note is affected, if two sequences are input, MIDI note and note length are affected, and if three sequences are provided, MIDI note, note length, and note velocity are all affected.



Example Command-Line Code:

```
bruce:oeis bruceadawson$ python3 oeis-search.py A000001 A000002 A000003 120.0
```

Figure 4.1: OEIS-to-MIDI Flowchart and Communications Diagram

4.3.3 Workflow

A typical use scenario begins with using command-line prompt to select up to three integer sequences from the OEIS database as well as tempo input (Example command-line prompt: python3 oeis-search.py A000010 A000056 A000200 120.0). The first integer sequence input specifies MIDI note values, the second integer sequence input specifies MIDI note velocities, and the third integer sequence input specifies note length. Tempo can be distinguished via command-line prompt, and if the user is using Ableton Live 9 DAW with the LiveOSC plugin, tempo information can be sent via OSC from the MIRKov system to Ableton Live.

Following command-line input, MIDI notes are then output through the local computer's IAC driver, using the integer sequence input to determine the MIDI note values, MIDI velocity values, and MIDI note length values. As integer sequences can have thousands of terms by nature, only the first five to thirty values of the integer sequences contained within the OEIS database and are output by the program.

4.4 Example Use Case of OEIS-to-MIDI

An example use-case of OEIS-to-MIDI using up to three integer sequences as an input source would output a MIDI melody according to the specified integer sequence(s) given. The following use-case (Figure 4.2) applies the following three integer sequences found in the OEIS database to melodic content: A117153 (MIDI Note), A000056 (MIDI Note Velocity), A0000010 (MIDI Note Length). Notes played, velocities of those notes, and lengths of those notes have been generated by the provided integer sequences.

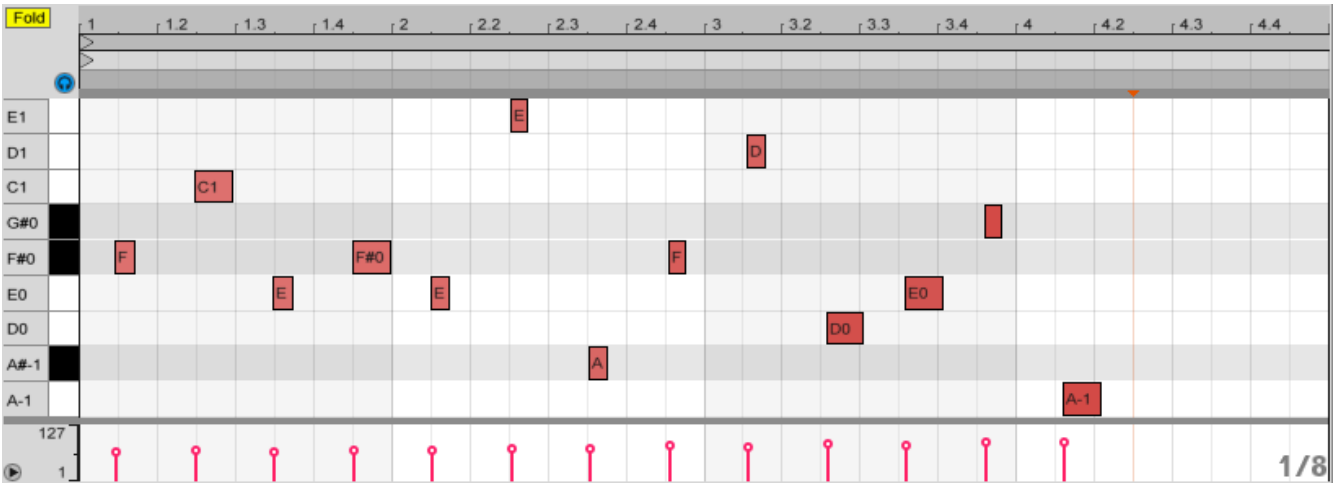


Figure 4.2: OEIS-to-MIDI Use-Case

4.5 Summary

This section describes OEIS-to-MIDI, a CAAC software tool and approach for generating computer-aided MIDI-based melodic content. Working with integer sequences and specifically the OEIS database, unique artistic possibilities are generated for musical output, accounting for a novel approach to CAAC methodologies. Not only can OEIS-to-MIDI create artistic possibilities within a CAAC-based compositional approach, but we hope that this project establishes a musical foundation for digital composers desiring the use of integer sequences and the artistic possibilities that arrive when using integer sequences in a CAAC-based approach. Additionally, development of this project will remain open-source and continue development using the C++ language to further assist system compatibility and system performance.

Chapter 5
LSysGenerator: Lindenmayer-System Chord Generator

"we may further establish that the role of the living composer seems to have evolved, on the one hand, to the one of inventing schemes (previously forms) and exploring the limits of these schemes, and on the other, to effectiving the scientific synthesis of the new synthesis of the new methods of construction and of sound emission."

– Xenakis, I. (1992)

This chapter presents the use of L-Systems within the domain of contemporary CAAC practices. The use of L-Systems is not new to the field of CAAC, however the advancement of technology has allowed the implementation of L-Systems into a user-friendly digital environment compatible with current Digital Audio Workstations (DAW). The use of L-Systems account for CAAC methodologies as using algorithmic means for string rewriting and propagation, which can be mapped to musical parameters in numerous user-defined ways. Select expansions of L-Systems are discussed to differentiate the uses and purpose of expanded L-Systems as an appropriate approach to CAAC methodologies.

5.1 Motivation

As technology's enhancements enable the creative mind to express thought through multimedia technology, increasingly complex models for organizing sound have been developed, and new models for organizing sound continue to be developed. Technological advancements are creative advancements for

the arts with higher-capacity computing allowing freedom for higher-capacity multimedia creative expression. Additionally, music has benefited from increasingly complex models for sound organization and as such experimentations in sound have incorporated influences from numerous fields, including but not limited to mathematics.

The motivation behind this project is that of using complex algorithmic systems for user-friendly real-time musical input and output. This project allows the end user, in accord with Ariza's seven descriptors of a CAAC system, to create musical chord progressions with the L-Systems string rewriting capabilities as a means for intelligent, theoretically related chord progressions. Additionally, this program is considered extensible, being available online in open-source format. In regard to using L-Systems as a mean for musical output via string rewriting, "An analytic grammar [such as L-Systems] is, basically, a parsing algorithm. An input sequence is analyzed according to its fitness to a structure, initially described by means of a generative grammar. (Stelios 2006)"

5.2 Background

The basic concept of L-Systems, and the numerous derivatives of an L-System, is that of string rewriting – rewriting given strings according to set parameters. "In 1968 Aristid Lindenmayer, a biologist, introduced a formal method for modeling the development of plants. Now called L-systems, Lindenmayer's method is a type of *rewriting system*, a general tool for constructing complex objects by starting with a simple object and recursively replacing parts according to instructions provided by a set of rewriting rules. (Frame 2015)"

The first systematic approach to string rewriting was by Mathematician Axel Thue (Stelios 2006). Additionally, Noam Chomsky further developed the basic idea of string rewriting for the purpose of linguistic models and formal grammar. The first published graphical representation of a L-System appeared in 1974, "produced by Frijters and Lindenmayer, and by Hogeweg and Hosper. The potential of L-systems for producing realistic images of plants was demonstrated in 1978 by Smith [and] In 1979 Szilard and Quiton showed L-systems can generate fractal curves (Frame 2015)."

Historically, a "Kolam," is a decoration design native to the villages of southern India (Frame 2015). Prusinkiewicz and Hanan's research proved that many intricate kolam designs can be generated with L-Systems (Frame 2015). The kolam design pattern is, "five millenia old, this artform is described in many ancient Sanskrit texts (Frame 2015)."

5.3 Lindenmayer Systems

In 1968, a Hungarian botanist and theoretical biologist at the University of Utrecht named Aristid Lindenmayer introduced a new string rewriting algorithm known as, "L-Systems." (Stelios 2006)

Difference between L-System and Chomsky generative grammar is that re-writing happens in a parallel fashion, rather than sequentially. "This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time. Parallel production application has an essential impact on the formal properties of rewriting systems. For example, there are languages which can be generated by context-free L-systems (called OL-systems) but not by context-free Chomsky grammars." (Prusinkiewicz, Lindenmayer, 1990)

5.4 Categories of Lindenmayer Systems

L-Systems are categorized by the type of grammar used, as each type of L-System, and each set of production rules is uniquely applicable in various ways, according to the type of L-System and type of rules applied. The Categories of Lindenmayer Systems consist of:

- Context-free (OL) Systems
- Context-Sensitive (IL) Systems
- Bracketed L-Systems
- Propagative (PL) Systems
- Non-Propagative L-Systems
- Table-based (TL) System
- Parametric L-System
- Deterministic, Context-free (DOL) System

5.4.1 Context-free (OL) Systems

Lindenmayer's original L-system was an OL-System created for the purpose of modeling algae growth. Context-free systems (Figure 5.1) represent the following form:

predecessor → successor

According to Stelios, the "*predecessor* is a symbol of alphabet V, and successor is a possibly empty word over V (Stelios 2006)."

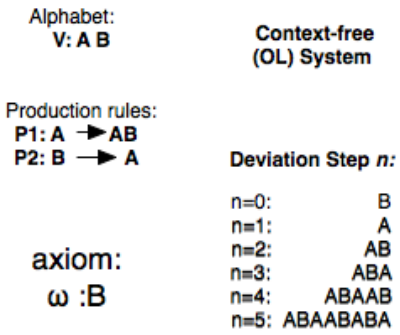


Figure 5.1: Context-free L-System

5.4.2 Context-Sensitive (IL) Systems

Application of the IL-system (Figure 5.2) may depend on the chosen context that the predecessor symbol appears within the system. This type of system is often used in the context of signal propagation simulation (Stelios 2006). Context-Sensitive systems represent the following form:

$$\text{left context} < \text{predecessor} > \varnothing \rightarrow \text{successor}$$

$$\varnothing < \text{predecessor} > \text{right context} \rightarrow \text{successor}$$

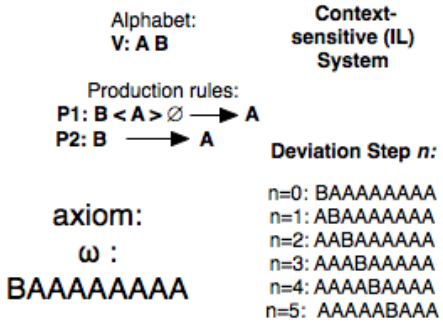


Figure 5.2: Context-Sensitive L-System

5.4.3 Bracketed L-Systems

The behaviors of the Bracketed L-Systems (Figure 5.3) allows the successor of a production rule to use one or more branching symbols, subsequently "branching," new axioms through each iteration of the system. The branching symbol, commonly a bracket or pair of brackets, "[ ]," allows the generation of an axiom at a new position inheriting the current position and other defined attributes within the system.

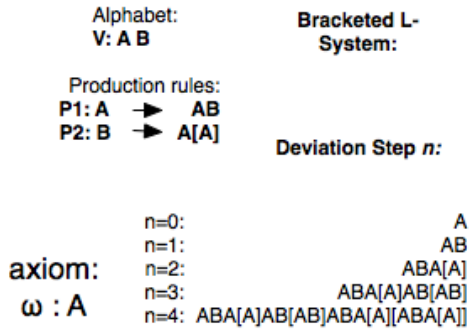


Figure 5.3: Bracketed L-System

5.4.4 Propagative (PL) Systems

PL-Systems measure the propagation of growth deviation length through each iteration of the systems mutation. According to Lindenmayer et al, "[PL-Systems are] a function that describes the number of symbols in a word in terms of its derivation length (Prusinkiewicz-Lindenmayer, 1990)." By analyzing the number of symbols in L-Systems, overlap can be found with known integer sequences such as the Fibonacci series, within the first five derivations of the OL-System above (1, 1, 2, 3, 5, 8, et cetera).

5.4.5 Non-Propagative L-Systems

During iterations of a Non-Propagative L-Systems, the length of the string is static and remains the same in length after each derivation. Non-Propagative L-Systems represent the following form:

$$\text{predecessor} \rightarrow \text{successor}$$

Results obtained from this type of L-System can yield results identical to those of a cellular automata algorithm, with method differing (Stelios 2006). Since the string length remains the same through each iteration, there is no data amplification or expansion of the string in this type of L-System.

5.4.6 Table-based (TL) Systems

The TL-System (Figure 5.4) is an extension of the L-System concept, allowing multiple "tables," of alternating production rules. This powerful expansion allows further mutation when allowing the parameters of the system to move between multiple tables – an external user-defined mechanism is used to chose which set of rules will be applied for the current derivation. This type of TL-System was introduced and formalized by G. Rozenberg to simulation the effect of environment during plant development (Stelios 2006).

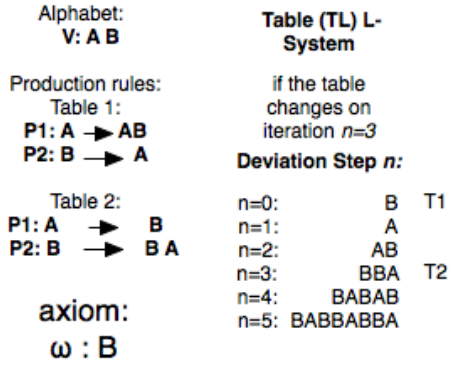


Figure 5.4: Table-based L-System

5.4.7 Parametric L-Systems

Parametric L-systems include, "parametric words," being strings consisting of symbols with associated parameters. The symbols belong to the alphabet **V**, similarly to the previously mentioned L-Systems, and the parameters of the Parametric L-System belong to the set of real numbers **R**. Parametric L-Systems provide a meaningful way of implementing control mechanisms for dealing with situations where irrational numbers need to be used.

5.4.8 Deterministic (DL) Systems

Both OL-Systems and IL-Systems, listed above, are considered deterministic systems. In a DL-System, "every word appears only once at the left of a production rule – meaning that a predecessor word is always replaced by the same successor (Stelios 2006)."

5.5 Application

The LSysGenerator program aims to account for the traditional method of string rewriting found in L-Systems, as an approach to CAAC methodologies. Creating this program allows users to use CAAC-based approaches to musical composition in a digital environment.

5.5.1 Design Overview

The LSysGenerator application, written in the chucK programming language, accounts for the string rewriting methods found within the L-System domain. The LSysGenerator program has been designed to include a graphical user interface (GUI), intended to be user-friendly and allowing for easy manipulation of parameters contained within the L-System. As such, the following list provides an overview of the main software requirements:

1. Support for a variety of MIDI input devices.
2. Graphical User Interface (GUI) for ease of access.
3. MIDI Output for use with external synthesizers, MIDI through devices, and DAWs.

5.5.2 System Overview

The general flow of this software is documented in Figure 5.5. A user provides an Axiom, or an initial string for the LSysGenerator application to rewrite. Parameters within the program are the manipulated to the user's liking, and unique musical output based on the Axiom and the rules of the L-System is created.

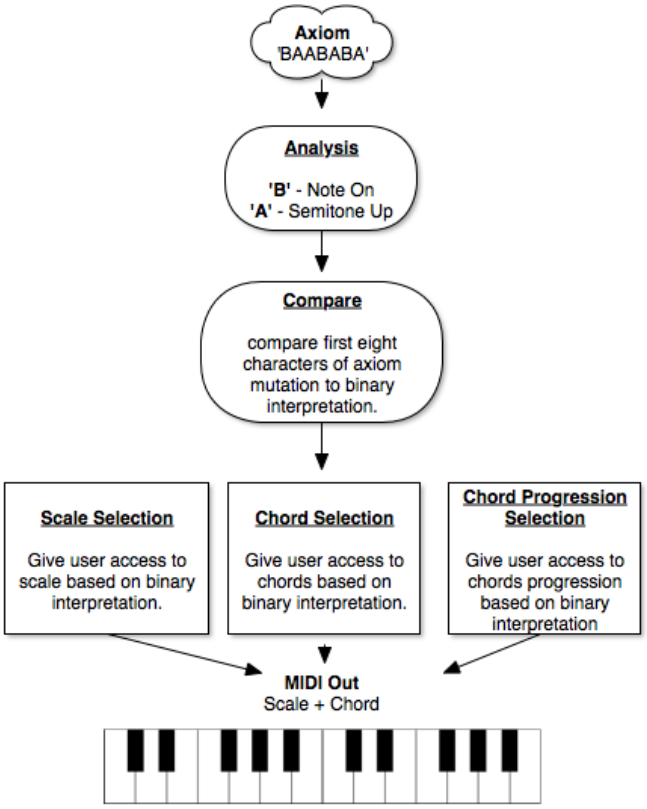


Figure 5.5: LSysGenerator Flowchart and Communication Diagram

5.5.3 Workflow

A typical use scenario begins with opening the project code with miniAudicle, an integrated development environment (IDE) for software development with the chucK programming language. At this time, using miniAudicle to open the code is the only publicly available way to display GUI elements within the chucK programming language. Using the GUI interface, axiom iteration depth can be determined, which then determines which musical scale, chords, and chord progressions will be output via the user's local IAC Bus. Additionally, user parameter control for manipulating the chord progression is available – allowing a user to "lock" a discovered chord progression, as well as move through the currently selected chord progression forward, backwards, forward-and-backwards, and random selection.

5.6 Example Use Case of LSysGenerator

An example use-case of the L-System Chord Generator would modify the specified, "axiom," and output scales and chords according to how the axiom was manipulated. Output varies depending on the specified

axiom, depth of axiom iterations, and current notes being played. Parameters are accessible to the user via a graphical user interface (GUI), and can also be mapped directly to external midi controls (Figure 5.7).

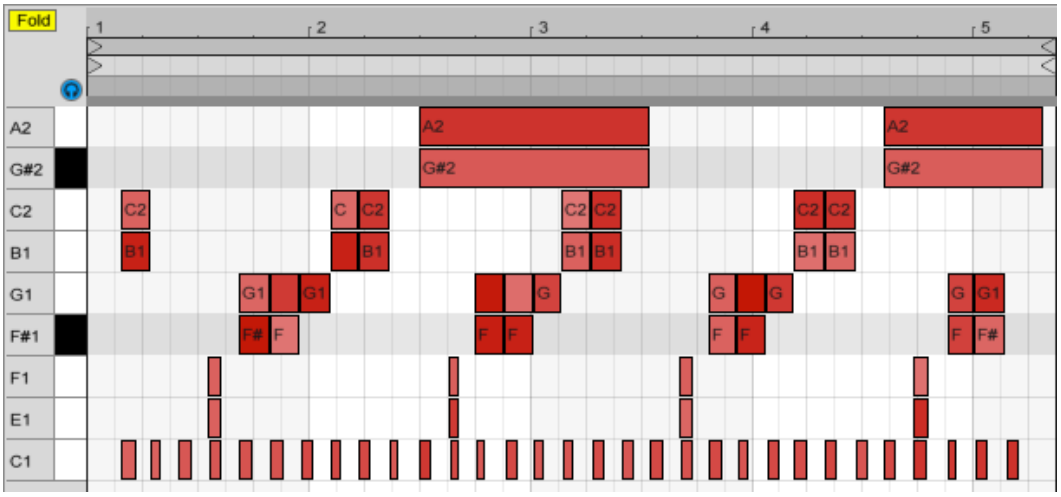


Figure 5.6: LSysGenerator MIDI Output

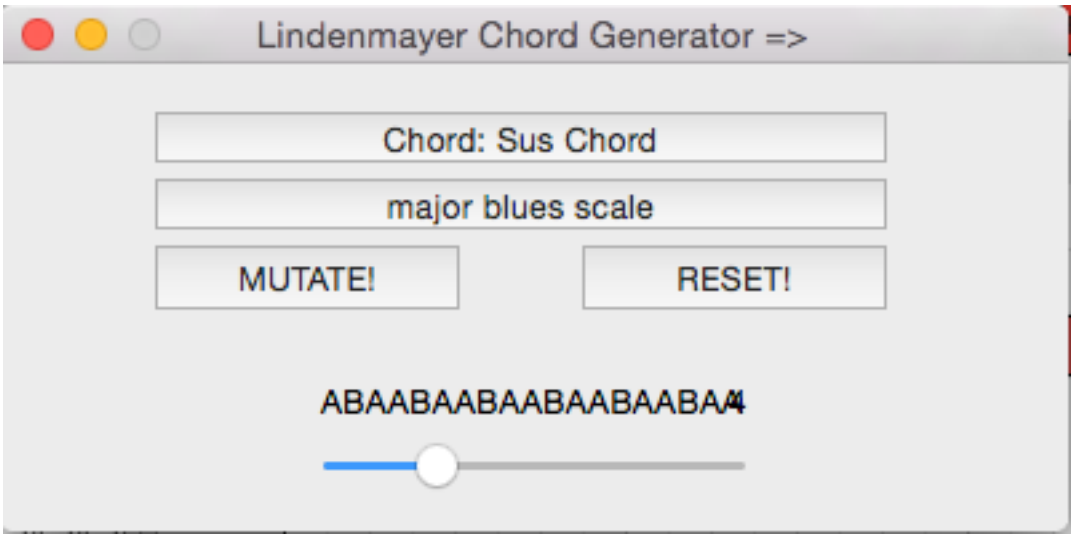


Figure 5.7: LSysGenerator GUI

5.7 Summary

This section described the LSysGenerator program, a CAAC software tool for generating computer-aided MIDI-based musical content in digital compositional environments. The Lindenmayer-System Chord

Generator is an adequate approach to CAAC because it utilizes the three paradigms of CAAC, computer-interaction, algorithmic-interaction, and human interaction. Working with L-Systems allows many unique musical possibilities from future applications of direct sound parameter manipulation to the current possibility and application of generating new musical content as an approach to CAAC based on the parameter manipulation found within the domain of L-Systems. L-Systems can enable musical discovery within the guidelines of musical composition – establishing a foundation for the use of L-Systems in within DAW environments, accounting as a viable compelling approach to CAAC methodologies. Additionally, development of this project will continue using the C++ language to further assist system compatibility and system performance.

Chapter 6
Conclusion

The overarching goal of this thesis has been to explore approaches to CAAC-based methodologies for music composition practices in a digital compositional environment. Throughout the duration of this exploration of CAAC methodologies, a few notable steps presented themselves:

1. Exploration of CAAC approaches must adhere to the definition of an "algorithm," as set forth by this thesis.
2. Musical parameters for algorithmic manipulation via CAAC methodologies must be clearly defined.
3. Algorithmic processes must be time-efficient and computationally practical for musical implementation.
4. Musical output may require compositional transposition, considering CAAC systems do not adhere directly to music theory principles, and rather, are a raw output intended for musical transposition.

6.1 Summary

This research identified methodologies to CAAC-based approaches for musical composition in a digital compositional environment. Examples were provided that demonstrated the viability of CAAC systems within the context of musical composition in a digital environment. To facilitate CAAC methodologies, this thesis developed valuable software tools that make it made it possible to use algorithmic processes for compositional purposes. As a result, this research generates many new possibilities for CAAC-based approaches to music composition in a digital environment, facilitating the discovery and implementation of unique musical possibilities.

6.2 Primary Contributions

The primary contributions of this research we believe will significantly add to future exploration of CAAC methodologies within digital compositional environments. The primary contributions of this research include:

1. Three new software tools that enable anyone to easily access and apply CAAC-based techniques within digital compositional environments, including:
    a. A new software tool for enables users to easily access MIR methodologies for the purpose of CAAC.
    b. A new software tool for extracting data from the OEIS database, allowing musical adaptation and transposition of integer sequences contained within.
    c. A new software tool for applying L-Systems to musical output, allowing manipulation of chords, scales, and progressions via CAAC methodologies.
2. A study of how CAAC techniques such as MIR. Integer sequences, and L-Systems can help lead to musical discovery and create unique artistic musical content based on this musical discovery.
3. Investigating into the future of CAAC methodologies within current digital compositional practices by creating tools that can be used regularly during the digital compositional process.

6.3 On Algorithmic Integrity

Musical focus is subjective – as some CAAC artists may focus on the raw output of the algorithmic processes, while others may transpose said output. Although CAAC techniques can be the subject of musical transposition, it is important to specify artistically how much transposition can jeopardize the integrity of the algorithmic output. While some algorithmically driven musical content may be transposed entirely using the algorithmic outputs as source material only, conversely other algorithmic-means of musical content may be analyzed it the raw output form. This is a factor to be determined solely by the user. Because CAAC methodologies can be mapped specifically to clearly defined musical parameters, CAAC systems are able to generate musical content on multiple levels, for e.g. (1) the note-level, (2) effect-level, and (3) score-level, (Wanderley et al., 2002). Respectively, note-level refers to musical content generated in context of notes, including but not limited to the velocity or strength of the note onset, as well as the duration and release of that note. Effect-level refers to musical content generated in context of effects and digital signal processing, pertaining to manipulation of parameters of effects pertaining to the musical content at hand.

The application flow of CAAC systems depends on the implementation of the specific algorithmic being used, with some adhering to principles of *constrained linear motion* (Wanderley et al., 2002), the response of a computer system to a particular user input, while others adhere to principles of *constrained circular motion* (Wanderley et al., 2002), another automated input-response for handling tasks.

6.4 Final Thoughts and the Future

Since the infusion of computers and algorithmic composition beginning in the 1960's, approaches to CAAC have taken on many algorithmic forms, exploring the domain of algorithmic composition and fueling the musical discovery process by generating source material for musical transposition. The beauty of CAAC lies within the realm of discovery and composition – between these two realms algorithmic processes reside, allowing compositional decisions by the composer, allowing algorithmic decisions by the computer, and creating opportunity for musical discovery that did not exist before the algorithmic process begun. Using technology as a tool for algorithmic advancement, the domain of algorithmic composition can continue advancement and the beauty of CAAC can continue to flourish.

The process of researching CAAC has been one of trial-and-error, a process of comprehending, learning, and relearning mathematical principle as much as it is comprehension and application of musical principle – in both the design of CAAC systems, and in functionality contained within. The domain of CAAC lies within the knowledge of algorithmic systems and within the knowledge of musical systems. Without adequate understanding of both domains, tools cannot be thoroughly utilized. It takes an authentic composer to utilize CAAC systems as much as it requires an authentic mathematician.

The future of exploration of CAAC methodologies is prime for composers and mathematicians alike as the integration between the domains of mathematics and music draws closer together through research within the CAAC domain and other technological advancement.

Bibliography

Ariza, Christopher. 2005. “An Open Design for Computer-Aided Algorithmic Computer Music: athenaCL.”

Ariza, Christopher. 2010. “athenaCL Tutorial Manual: Third Edition, Version 2.0.0a15.”

Buxton, William. 1977. “A Composer’s Introduction to Computer Music.” *Interface* 6: 57–72.

Basharin, G.P., A.N. Langville, and V.A. Naumov. 2004. “The Life and Work of A. A. Markov.” Http://www.maths.tcd.ie/~nhb/talks/Markov.pdf.

Brown, Judith C. 1990. “Calculation of a Constant Q Spectral Transform”. MIT. http://140.127.114.187/download/%E5%BB%BA%E5%BE%B7%E7%A0%94%E7%A9%B6%E6%89%80%E8%B3%87%E6%96%99/Spectrograms%20Constant-Q%20(Log-frequency)%20and%20conventional%20(linear)/Calculation%20of%20a%20constant%20Q%20spectral%20transform.pdf.

Cano, Pedro E. 2002. “A Review of Algorithms for Audio Fingerprinting”. IEEE. http://ucbmgm.googlecode.com/svn-history/r7/trunk/Documentos/Fingerprint-Cano.pdf.

Casey, M.A., R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. “Content-Based Music Information Retrieval: Current Directions and Future Challenges.” http://doc.gold.ac.uk/~mas01cr/papers/procieee2008/cbmir.pdf.

Copeland, Jack B. 2004. *"Colossus: Its Origins and Originators*. IEEE Annals of the History of Computing 26, no. 4.

Courtney, David. 2015. “Chandrakantha Music of India.” http://chandrakantha.com/faq/tala_thalam.html.

DiCarlo, Christopher. 2001. “Interview with Maynard James Keenan.”

Devlin, Liam. 2002. “Athanasius Kircher, Musurgia Universalis”, November. http://special.lib.gla.ac.uk/exhibns/month/nov2002.html.

Doornbusch, Paul. 2004. “Computer Sound Synthesis in 1951: The Music of CSIRAC”. Computer Music Journal 28.

Downie, J.S. 2004. “The Scientific Evaluation of Music Information Retrieval Systems: Foundations and Future.” *Computer Music Journal* 28 (2): 12–23.

Edwards, Michael. 2011. “Algorithmic Composition: Computational Thinking in Music.” *Communications of the ACM* 54 (7): 58–67.

Flanagan, JL. 1972. “Speech Analysis, Synthesis and Perception.” *Springer- Verlag, New York*.

Frame, Michael. 2015. “Fractal Geometry.” http://classes.yale.edu/fractals/.

Hollerweger, Florian. 2015. “Music and Technology - Sound Design Syllabus”. MIT. http://ocw.mit.edu/courses/music-and-theater-arts/21m-380-music-and-technology-algorithmic-and-generative-music-spring-2010/lecture-notes/MIT21M_380S10_lec07.pdf.

Knuth, Donald E. 1973. “The Art of Computer Programming. Vol. 1: Fundamental Algorithms. Vol. 2: Seminumerical Algorithms.” Reading, Mass.: Addison-Wesley.

Koenig, G. M. 1971. “The Use of Computer Programs in Creating Music”. Paris: La Revue Musicale. http://www.koenigproject.nl/Computer_in_Creating_Music.pdf.

Loy, Gareth. 2006. *Musimathics: The Mathematical Foundation of Music*. The MIT Press.

Manousakis, Stelios. 2006. “Musical L-Systems”. The Royal Conservatory, The Hague.

Meurer, William. 2004. “Aspects of Music Information Retrieval”. The University of Texas. https://www.ischool.utexas.edu/~i385df04/StudentPPT/HTML/meurer_w/AspectsofMIR.pdf.

Nierhaus, Gerhard. 2001. *Algorithmic Composition: Paradigms of Automated Music Generation*. SpringerWeinNewYork.

Nettl, Bruno. 2015. “Tala.” *Britannica*. http://www.britannica.com/EBchecked/topic/581274/tala.

Nguyen, Hieu D. 2011. "Mathematics by Experiment: Exploring Patterns of Integer Sequences."
O'Neill, Christopher. n.d. "Fibonacci."
http://www.math.rutgers.edu/~cherlin/History/Papers1999/oneill.html.
Pauws, Steffen. 2004. "Musical Key Extraction from Audio". Universitat Pompeu Fabra.
Pfeiffer, Silvia, Stephan Fischer, and Wolfgang Efelsberg. 1997. "Automatic Audio Content Analysis". University of Mannheim.
http://www.researchgate.net/profile/Silvia_Pfeiffer/publication/221572403_Automatic_Audio_Content_Analysis/links/09e4150699a8e6d2c9000000.pdf.
Prusinkiewicz, Przemyslaw, and Aristid Lindenmayer. 1990. "The Algorithmic Beauty of Plants." New York: Springer-Verlag.
Roads, Curtis. 1996. "The Computer Music Tutorial". Cambridge: MIT Press.
Swanson, Christopher. n.d. "Guido D'Arezzo Italian Benedictine Monk". Longwood University.
http://www.longwood.edu/staff/swansoncl/Sightsinging/GUIDO%20AREZZO.htm.
Smith, Julius O. 2011. *Spectral Audio Signal Processing*.
Smith, Peter F. 2003. "The Dynamics of Delight: Architecture and Aesthetics."
Temperley, David. 2007. *Music and Probability*. The MIT Press.
The OEIS Foundation, Inc. n.d. "Brief History." http://oeis.org/wiki/Welcome#OEIS:_Brief_History.
Van Loan, Charles. 1992. *Computational Frameworks for the Fast Fourier Transform*. SIAM.
Wang, Avery Li-Chun. 2003. "An Industrial-Strength Audio Search Algorithm". ISMIR.
http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf.
Wanderley*, M.M., and Nicola Orio. 2002. "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI". McGill University. https://blog.itu.dk/DDAA-E2013/files/2013/09/wanderley-evaluation-of-input-devices.pdf.
Wang, Ge, and Perry Cook. 2003. "ChucK: A Concurrent, on-the-Fly Audio Programming Language." Proceedings of International Computer Music Conference.
WIRED Ventures Ltd. 1994. "Digital Expression." http://web.media.mit.edu/~nicholas/Wired/WIRED2-12.html.
Wright, Matthew. 1997. "Open Sound Control-a New Protocol for Communicationg with Sound Synthesizers."
http://cnmat.berkeley.edu/system/files/attachments/S1355771805000932a.pdf.
Xenakis, Iannis. 1992. Formalized Music.