A Computational Approach to the Syntax and Semantics of Music

Halley Young

Advisor: Bud Mishra

5/10/2017

Table of Contents

Introduction

Analysis of music is nearly as universal a feature of cultural life as is music itself. Throughout written history, music has been described as something inherently connected to the physical world around us, as a source of mathematical truth, and as akin to human language. While there is much to say about all three of these views, this paper is largely concerned with a marriage of the latter two – namely, how to formally describe the relationship between music and language. Reviews of models of various aspects of music will be interspersed with descriptions of actual implementations of music generation based on these models. I include and critique models that have already been developed by other researchers, and build new formalisms as well.

This paper will include the following sections: In the first section, I will discuss the history of musical analysis by music analysts, composers, and biologists. I will then pose the question to be addressed throughout the paper – what could the words "syntax" and "semantics" possibly mean when applied to music? In order to begin to answer this question, I will first provide a basic description of the most common linguistic formalisms used by syntactitians and semanticists. I then present what is known about musical universals as opposed to dominant musical repertoires.

In the following sections, I describe various approaches that have been taken by me and others towards this problem. I discuss these approaches roughly in the order of how viable I believe they are, with the less promising approaches described first. I first present and implement music using one of the oldest tools in music cognition and automatic music generation to discuss music: simple probabilistic processes (which I argue are interesting but too simplistic to provide a foundation for musical semantics). I then discuss several different uses of generative grammars as they have previously been defined to describe music (and discuss the results of attempting to translate these

approaches into generative algorithms). I proceed, starting with a crude attempt by others to describe musical state space, to develop my own theories of what musical state space could look like, and discuss my experiments creating a database of musical points in state space. I present a musical type theory and a method for generating music based on program synthesis. I explicitly compare this method to the formal methods I described earlier that linguistic semanticists have used, showing that there is a significant correspondence.

In the final sections of my paper, I address several points which are not central to my thesis, but are likely to be on computational musicologists' radar and are worth discussing to better understand the field of musical semantics as a whole. I bring up theories that have been cited as foundations of music theory, and explain why I do not see them as such. I discuss current research on using neural networks to generate music, as well as my take on how they should and should not be used in the future. I also introduce examples of repertoire-specific semiotic approaches to tonal music that can be applied to generating very limited musical outputs. Finally, I discuss special cases in which music can be said to have semantics beyond what is normally perceived: namely, in film music, vocal music, and data sonification.

An Introduction to the Problem

Between Math, Language, and Art: A Brief History of Musicology in the Western World

Music analysis in the Western World can be traced back at least twenty five hundred years, to when Pythagoras discovered that there is a direct relationship between the properties of sounding materials and the perception of tone: an anvil struck against a tube 2 units long will produce an interval of a fifth against a tube 3 units long.[1] To Pythagoras and his followers, this discovery suggested the equivalence of "musica humana," or the order defining the human body and brain, and "musica universalis", or the order defining the physical world. It also suggested that there were certain intervals that were more "perfect" and should be the foundation of music – namely, intervals associated with proportions which could be reduced to small numerators and denominators.

Through the medieval ages, scholars continued in the vein of Pythagoras to pursue "musica speculativa," looking for mathematical relationships in music which would suggest what music should and shouldn't be like, and which would be directly related to the order of the divine cosmos. As such, music was part of the "quadrivium," (the set of sciences that each student was to learn), as opposed to the "trivium" (the set of arts).[2] However, towards the end of the Renaissance the role of music was reevaluated.[3] Music came to be understood in two new ways which are relevant to this paper: Music became known as a tool to express and induce emotional states. The way in which this was done was through "musica poetica," in which the classical latin-oriented techniques for rhetorical devices were

---

1 Bartel, Dietrich. (1997) *Musica Poetica.* University of Nebraska Press.

2 IBID

3 IBID

modified to be used as musical devices. For instance, "anaphora," a standard tool in rhetorical speaking in which a phrase is repeated at the beginning of several sections, was translated into a musical device in which a part of a phrase was repeated. These developments (the use of music as a description of emotional states and as heightened speaking) will be discussed later in this paper when I consider what a semantics of music could possibly mean. It is also relevant to the history of music cognition that with the description of music as affection-inducing came very specific theories about the way in which music induced changes in bodily humors and in the soul (which was understood to be a physical entity located in the pineal gland).

The late eighteenth and nineteenth centuries saw several new developments in the appraisal of music, as well as some controversy. According to Solomon, Beethoven was the first notable musician to consider himself an "artist" in the modern sense as opposed to a craftsman.[4] In the nineteenth centuries, the idea of the artist morphed into that of an individual who was creating displays of his own heightened emotions that would not be able to be displayed otherwise (which differed from the earlier understanding of music as creating displays of objective emotion). This new understanding was not without controversy – while philosophers such as Schopenhaur enthused music's status as an unparalleled "emotional language,"[5] other art critics such as Hanslick remained convinced that music was a formal game to be played by manipulating structures rather than emotional states.[6]

Modern Western music theory (early twentieth century onwards) is fundamentally different from any preceding era. The nineteenth century ideal of expressing oneself in a unique way led to a

---

4 Solomon, Maynard. (2004) *Late Beethoven: Music, Thought, Imagination.* University of California Press.

5 Dunton, Green. (1930). Schopenhauer and Music. *The Music Quarterly,* Vol 16, pp 199-206.

6 Kimmey, Roy. (2009). The Critic in Question: Eduard Hanslick and Viennese Musical Identity. *The Harvard College History Review*, Vol X, Issue 1.

new musical aesthetic in which what was prized was largely being different than previous works. Composers enthusiastically designed compositional systems which destroyed long-lasting musical structures: For instance, serialism was introduced by Schoenberg to circumvent the system of tonality.[7] Serialist music sounds to most listeners with a Western background as unpatterned, cacophonic, and chaotic, which was hailed as a virtue. Since then, composers have been constantly inventing mechanisms for creating music which is not like anything seen before. As such, both composers and non-composing theorists have been developing new formalisms – composers to find a strong basis for composition, and theorists to wrestle with individual works which cannot be analyzed by traditional means. As composers have developed individual methods for composition, the number of formalisms needed has grown exponentially – a model for dealing with post-tonal canon might fit one piece,[8] while a method for canons in augmentation might fit another.[9] Some formalisms have become increasingly mathematical – for instance, the application of group theory to post-tonal music is now widely accepted.[10]

In addition to changes in music analysis and composition, the twentieth century saw the development of the study of music cognition. While, as mentioned above, considering music's effects on the body was far from new, with the Cognitive Revolution scientists now had the chance to make testable predictions about how music is perceived and reacted to in the brain. As will be discussed below, cognitive scientists have asked basic questions: Is music biological in nature? How does our

---

7 Simms, Bryan. (2000) *The Atonal Music of Arnold Schoenberg 1908-1923.* Oxford University Press.

8 Morris, Robert. (2004-2005). Aspects of Post-Tonal Canon Systems. *Integral*, Vol 18-19, pp 189-221.

9 Van der Walt, Salome. (2007). *Rhythmic Techniques in A Selection of Oliver Merriaen's Piano Works* (master's thesis). Retrieved from http://www.repository.up.ac.za/

10 Zhang, Ada. (2009). *The Framework of Music Theory as Represented With Groups.* Retrieved from https://sites.math.washington.edu/~morrow/336_09/papers/Ada.pdf.

brain turn sound waves into structured information? How are we changed after listening to a piece of music?

Equally important to this paper is the effect of the Cognitive Revolution on the study of language, which was then translated into new research in musicology. Noam Chomsky introduced the use of formal grammars to describe language, and in doing so explicitly connected mathematical and computational models of certain structure with actual language. He also created ways of describing and categorizing formal languages.[11] In the 1960's Richard Montague developed a theory of semantics, rigorously defining what it means for a word to "mean" anything. These works were connected to music in 1973, when Leonard Bernstein delivered six famous lectures dedicated to exploring the relationship between linguistic and musical structure, including phonology, syntax, and semantics.[12] His work was continued by musician Fred Lerdahl and linguist Ray Jackendoff, who together came up with a linguistic framework for describing first tonal and then post-tonal music (described in greater detail below).[13]

What Could "Syntax and Semantics of Music" Mean?

As the above summary of Western music history indicates, throughout the past two thousand years, music theorists have discussed music as something akin to language, mathematics, and biological stimulants. Very recently, musicians and linguists have begun formalizing these intuitions in terms of researching the syntax and semantics of music, and relating the existence of such structures to music's biological nature. No consensus has been reached as to what such research entails, much less

---

[11] Jiang, Tao; Li, Ming; Ravikumar, Bala; and Regan, Kenneth. *Formal Grammars and Languages.* Retrieved from http://www.cs.ucr.edu/~jiang/cs215/tao-new.pdf

[12] Bernstein, Leonard. (1973). *The Unanswered Question: Six Talks at Harvard*. Cambridge, Massachusetts: Harvard University Press.

[13] Lerdahl, Fred and Jackendoff, Ray. (1981). *A Generative Theory of Tonal Music.* The MIT Press.

what the answer is. Thus, it makes sense to enumerate all of the possibilities of what a syntax or semantics of music could be, and to set up some basic requirements that any possible solution must have in order to be viable.

Syntax is slightly more straightforward to define than semantics, so I will start there. Tao Jiang et al define a grammar as a quadruple (Σ, V, S, P) where:

1. Σ is a finite nonempty set called the terminal alphabet. The elements of Σ are called the terminals.

2. V is a finite nonempty set disjoint from Σ. The elements of V are called the nonterminals or variables.

3. $S \in V$ is a distinguished nonterminal called the start symbol.

4. P is a finite set of productions (or rules) of the form $\alpha \rightarrow \beta$ where $\alpha \in (\Sigma \cup V)^* V (\Sigma \cup V)^*$ and $\beta \in (\Sigma \cup V)^*$, i.e. α is a string of terminals and nonterminals containing at least one nonterminal and β is a string of terminals and nonterminals.

To put it concisely, a grammar is thus a set of structure-building rules which describe the smallest bits the structure can be made out of and the ways in which they can be built up into larger structures. This definition can be applied at least in a superficial fashion to music – music is by definition the combination of individual tones into melodies, and there is at least some cognitive process or "rule" which must help composers perform this combination, as most composers are not leaving the notes to chance. However, whether there is a stronger hypothesis to be made about the grammatical nature of music is up to debate. Some music theorists see all of tonality as a syntactic formulation, in which the

role of individual chords contributes to the global coherence of the melody.14 While this interpretation is plausible, it only applies to Western music in the seventeenth to nineteenth centuries, leaving open the question of whether musical syntax is universal.

Semantics is even harder to define in relation to music than syntax. It is likely that much of the disagreement regarding whether music "has meaning" stems from a disagreement about what meaning is. One could, for instance, determine whether or not music has semantics solely based on whether the tools used by linguists to describe linguistic meaning, namely predicate logic, model theory, and lambda calculus, can be co-opted by musicologists. One could also take a more general approach, and ask whether any change has been made in the brain of the listener. Semantics can be hard to distinguish from syntax: as mentioned above, tonality can be understood as a syntactic structure, but surely the difference between major and minor chords can also be understood semantically.

Another way to approach the question of syntax and semantics in music is to discuss whether biologically music is processed similarly to the biological processing of language (to the extent that we understand how language is processed). Surprisingly, this approach does not provide much of an alternative to formal methods. First of all, evidence from fMRI studies is complex and contradictory. The standard theory is that, while musical and linguistic syntax is processed in a similar place, linguistic and musical semantics are processed differently.15 However, one study shows that musical semantics, defined as mood, use of leitmotifs, tension, and extra-musical allusions, overlapped in brain region with linguistic semantics.16

---

14 *Harmonic Syntax – The Idealized Phrase,* retrieved from http://openmusictheory.com/harmonicSyntax1.html

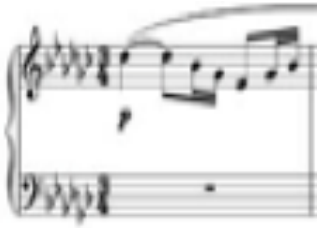
15 Kunert R, Willems RM, Casasanto D, Patel AD, Hagoort P (2015). Music and Language Syntax Interact in Broca's Area: An fMRI Study. *PLoS ONE* 10(11): e0141069. doi:10.1371/journal.pone.0141069

16 Koelsch. (2005). Neural substrates of processing syntax and semantics in music. *Curr Opin Neurobiol.* Vol 207-12

In addition, it is not possible to completely separate the question of what formalisms can describe music and what biological processes occur when music is heard or conceived of, as to talk about the biological nature of language and music one must necessarily invoke both the structure and functions of language and music. For instance, Geraint Wiggins, in his biologically-oriented piece "Syntax, music, and the meaning of meaning," introduces five properties that he argues apply to both music and language in order to suggest that they are processed similarly and play a similar function.17 These five properties include semiotics, conventionality, repetition with variation, structure, and hierarchy – essentially, the properties of having a syntax and semantics.

What must a theory of musical syntax/semantics do?

There are several properties of music that I believe any "theory of everything" of musical syntax or semantics must account for. The first property that must be accounted for is very well-documented: different aspects of music produce different reactions. Minor chords and slow rhythms (in Western audiences) evoke images of sadness, while major chords and fast rhythms evoke images of happiness. The second is much less understood. We usually hear more facets of music than there are notes. Just listening to the below snippet of piano music by Debussy



17 Wiggins, Geraint. (1998). Syntax, music, and the meaning of meaning. *Proceedings of the First Symposium on Music and Computers.* pp. 18–23.

I hear a pentatonic scale, a repetitive quarter-eighth-eighth motif, a complete motif inversion, an arpeggiation, an Eb7 chord, a stressed high note that has a pitch class of 2, etc. Any theory that can only explain one of these properties to the exclusion of others has not adequately described this bar of music. One would imagine this would be obvious; however, in fact, there are many theorists who view understanding the system of tonality and/or rhythm as the essence of understanding music. This is not to say that theories about how tonality works are not useful. Rather, I am claiming that they cannot be the sole foundation of musical semantics.

Musical Universals and Cultural Artifacts

Finally, before diving into theories of musical semantics, it is necessary to distinguish between what constitutes "musical sound" rather than "non-musical sound." Of course, this is a very difficult question with no one answer when considered from a philosophical point of view. Therefore, I shall attempt to address the definition of music by considering what music is actually found in world cultures. I will first consider the aspects of music found in most musical cultures, and then discuss how Western music fits with these universalities.

In 2011, Brown and Jordania compiled a list of aspects of music that are fairly universal.18 In the first part of their paper, they presented a list of contexts and behaviors relating to music. While these are not able to be easily directly translated into formal models of music, several are worth considering as we study how music works. Many cultures have different types of music for different occasions or purposes. There are often extra-musical associations of the various facets of music. Music is often used to induce emotions.

---

18 Brown S, and Jordania J. (2011) Universals in the world's musics. *Psychology of Music.* Vol 41(2), pp. 229-248.

More relevant to this paper are the expressive universals in music. Music is almost always composed of discrete pitches rather than continuous sound (a fact which is basic but also allows for conceiving of a musical “alphabet”). Octave equivalence, or the idea that two notes at a ratio of 2:1 are essentially the same, is both a universal truth and the foundation of many group theoretic treatments of music. Most types of music have rules regarding combinations of pitches into a scale, normally of eight or fewer pitches. Most have the idea of a tonic, or starting note. Most include precise and repetitive rhythms. Perhaps most important are the structural universals: Music is typically divided into phrases. Motifs, or small melodic riffs, are combined into larger melodies. Music is often composed of several different parts with different properties happening simultaneously. Music typically has a discernible beginning, middle, and end.

One can ask where Western music fits into musical universals. It is true that Western music is an outlier for several reasons. The development of a system of musical notation allowed Western music to become exponentially more complex than other musics, which had to be recalled through memory alone. Classical music is much more polyphonic (multi-voiced) than most world musics. Serialism, the technique introduced by Schoenberg to deliberately break all preconceived notions of what music should sound like, naturally violates many music universals: It consist of scales of 12 rather than less than eight notes, and it doesn't involve a tonic or home note. However, tonal music of the sixteenth to nineteenth centuries, while complex, actually fits a lot of the musical universals: it revolves around a scale of seven notes with a tonic, is grouped into phrases, and has well-defined sections.

Formal Models of Musical Syntax and Semantics

Simplistic but cognitively approximately correct: Probabilistic approaches to musical syntax and semantics

As mentioned above, musical cultures have conventions which govern how composers write and listeners perceive of music. Some of these conventions are fairly universal – one would more typically expect a small interval than a large interval in most, if not all, cultures. Others are more culture-specific; for instance, only a listener exposed to much Western music would expect a perfect cadence at the end of a phrase. Whether culture-specific or universal, and whether melodic, harmonic, or rhythmic, the idea of musical expectations can be modeled using probabilities. We expect high-probability events, and are surprised by low-probability events.

In "Music and Probability," Temperley describes how probabilistic processes govern musical perception.19 The fundamental insight involved in his research was Bayesian reasoning – that music has a surface and a structure, and the probabilities of different structures could be inferred from the probabilities of the surfaces. He proposes a model for how people track metrical and key structure throughout a piece of music. In doing so, he implicitly creates models for how surprising music is. Such probabilistic models have had the most success among cognitive musicologists who are interested in modeling actual human perception. Depending on one's level of cynicism, one can offer two explanations for its success in this field – that people's perception of music is really based on simple cognitive processes, or that the field of cognitive musicology has yet to discover how to deal with more complex musical information. There certainly is reason to believe the former: trials with human

---

19 Temperly, David. (2006). *Music and Probability.* The MIT Press.

subjects suggest that at least some probabilistic models can describe some aspects of human perception reasonably well.[20]

In addition to probabilistic perceptual models, probabilistic models are used to generate music. The most common device used to describe the probabilities of future events is the Markov model. An n-state Markov model specifies the probabilities of each current possibility, given n prior states. According to harmonic practice, a 3-state Markov model that models chord progressions which has a I-IV-V sequence as its prior inputs will assign a high probability to a I as the next output, and a low probability to a VII as the next output (this is a probabilistic fact about tonal music). A 1-state Markov model which has an A4 pitch as its prior state will have a high probability of a B4 pitch (a near pitch) as the next pitch, and a low probability of a G7 pitch (a very far pitch) as the next pitch.

How does this all relate to syntax and semantics? The significance of structure and surface in music hints at a comparison to the syntax of natural language: linguists talk about the surface being the words, and the structure being the grammatical rules that generate the string of words. In that sense, meter and tonality are the syntax that governs placement of notes in the alphabet of music. Furthermore, Markov models are also used in generating natural language. While Markov models are typically not used in formal semantics, they do play a role in corpus semantics, or work on extracting meaning from a body of text. (Mostly, this is work done by Natural Language Processing researchers who are looking for practical insight rather than by pure linguists.) Thus, Markov models of music can be said to explain something about semantics, even if exactly what is unclear.

Unfortunately, I do not believe that such simplistic probabilistic models can form a basis of knowledge of music. In practice, I have found that music based solely on Markov models seems to

---

20 Pearce MT and Wiggins GA. (2007). Evaluating cognitive models of musical composition. *Proceedings of the 4th International Joint Workshop on Computational Creativity*. pp 73-80.

wander randomly, with a lack of global coherence (one can hear an example the author generated of Markov music here: https://soundcloud.com/user-332259139/markov-model-music). This is because music is extremely context-sensitive. Probabilistic models work well when it is useful to know that in 90% of cases with respect to one variable, a certain output was chosen; however, in music one generally wants to consider so many variables that it is not possible to construct a table outlining all possibilities. Perhaps more detrimentally, Markov models assume a model of creativity in which one always precedes from a previously given corpus of works. This assumption is inviable for several reasons: It tells us nothing about music's origins (obviously there was a "first" piece of music which could not have been generated from previous examples, and it ignores acts of "historical creativity" in which artists have developed fundamentally new styles.21

Application of Generative Grammars to Music

Over the past 50 years, there have been several attempts to describe music in terms of the types of generative grammars that linguists use in working with natural language. In his Six Talks at Harvard, Leonard Bernstein introduced musician's to Chomsky's work in generative grammars, and speculated about their application to music. While Bernstein considered ways of analyzing musical figures of speech (much like the "musica poetica" of the Renaissance and Baroque ages), he did not put forward any formal way of comparing music to language. However, inspired by these talks, in 1963 musician Fred Lerdahl and linguist Ray Jackendoff developed a "Generative Theory of Tonal Music."[22] As the authors state, the theory provides a "formal description of the musical intuitions of a listener who is

---

21 Draper, Steve. (2015). *Creativity.* University of Glasgow. Retrieved from http://www.psy.gla.ac.uk/~steve/best/creative.html.

22 See Lerdal and Jackendoff, 1981.

experienced in a musical idiom." While it is not phrased in terms of context-free or context-sensitive formal grammars, a cursory inspection reveals that it actually consists of these ideas.

The authors asserted that music could be divided up into hierarchical units. This was not new; however, the authors expanded upon the nature of these hierarchies and how they are perceived. According to Lerdahl and Jackendoff, hierarchical structures can be perceived through grouping structure and metrical structure. Grouping structure can refer to the classical segmentation of large pieces of music into periods, periods into phrases, phrases into motifs, motifs into figures; it can also refer to forms not typically seen in classical Western music. At the same time, one perceives of metrical structure that is less connected to the content or self-similarity of a chunk of music and is more connected to the tempo, or basic beat. For instance, (taking a structure common to 18th century sonatas) a phrase could be broken into an antecedent and consequent, with the consequent broken into two motifs and the first motif composed of two figures; this is the grouping structure. The fact that this first motif is understood to be one bar with four regular beats (including beats that may not actually have a vocalized onset) is a reflection of the metrical structure.

As the theory stipulates, these structures both are induced by and induce two different types of reduction. A reduction is somewhat akin to the development of the term “NP” (noun phrase) to categorize the sequence “Det N” (determiner, noun phrase) in a phrase structure tree – the idea being that there is a different way to categorize what is happening on each different level of analysis. According to the authors, there are two types of reductions – time-span reductions and prolongational

reductions. Time-span reductions assign an “importance level” to each element in the structural tree.

These importance levels can be based on musical factors such as dynamics, rhythmic emphasis, or registral emphasis. On the other hand, prolongational reductions attribute “tensing” or “relaxing” properties to each structural item, determining whether they are propelling forward into the next structure or causing a degree of closure. This also can be based on a variety of musical factors, including rhythmic and tonal stability.

Lerdahl and Jackendoff's theory has certainly asserted its place in the study of musical syntax: the authors gave a reasonable description of what “structure” means when talking about music, as well as some of the properties of a given musical structure that have to be considered in order to assume an understanding of that structure. However, I believe it is also somewhat problematic. Surely there are properties of music, such as form, that do not straightforwardly correspond to closure or non-closure (at the very least, I'd like to see evidence that these properties do in fact contribute to closure or lack thereof). Are these properties not relevant to the syntax and semantics of music? What happens when meter is not straightforwardly reducible to hierarchical units, as is the case in much of the music in Africa? While these challenges are not insurmountable, it is also true that Lerdahl and Jackendoff's theory does not tell us nearly enough. What more refined features than “rhythmic stability” can be considered musical properties which affect perception? If we're comparing music to language, shouldn't we consider a “compositional semantics” in which a combination of properties together can reduce differently than each alone?

In addition to Lerdahl and Jackendoff's work on developing a more general theory of music loosely based on formal grammars, there have been attempts to create more specific theories of music that only describe eighteenth-century harmony. Quick devised “probabilistic temporal graph

grammars," in which larger time spans based on a certain chord would be broken down into individual stretches of related chords according to harmonic rules.[23] For instance, consider the below rule:

$$I_t \rightarrow I_{0.25t}\ IV_{0.25t}\ V_{0.25t}\ I_{0.25t}$$

This rule states that if one wants to compose a piece of music of length *t* that is in the chord-area of *I* (the tonic, or home note), one can create a sequence of a piece of music of length *0.25*t* in chord-area *I,* a piece of music of length *0.25*t* in chord-area *IV*, a piece of music of length *0.25*t* in chord-area *V,* and a piece of music of length *0.25*t* in chord-area I (this is a generalization of a perfect authentic cadence, a common musical object). This rule is one particular rule among several rules for breaking down a *I* chord-area, and a given rule is chosen probabilistically. Once this rule was implemented, one could keep the $IV_{0.25t}$ area as a single *IV* chord of length *0.25*t*, or could break it down further according to the rules associated with *IV*, which are of a similar nature as the rules associated with *I.*

Wilding took a different approach, analyzing music using the lambda calculus. He considered that sequences of chords are not orthogonal elements like letters in the alphabet, but rather have defined and (in tonal music) a small number of relationships to each other. For instance, one commonly sees in tonal music root motion by a fifth – a *I* chord becomes a *V* chord, a *V* chord becomes a *II* chord, etc. Alternatively, there are many root motions by fourth: a *I* becomes a *IV*, a *V* becomes a *I*, etc. Wilding describes these two relationships as functions: *right-onto*, and *left-onto.* He then defines chord progressions in terms of function applications: *I V I* becomes *left-onto(right-onto(I))*.24 This is similar to categorial grammars used by linguistics syntacticians and semanticist, in which each word

---

[23] Quick, Donya and Hudak, Paul. (2013). Grammar-Based Automated Music Composition in Haskell. *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design,* pp 59-70.

24 Wilding, MT. (2013). *Harmonic Analysis of Music Using Combinatory Categorial Grammar* (doctoral thesis). Retrieved from https://www.era.lib.ed.ac.uk/handle/1842/8019

corresponds to a lambda-term which when combined produces an evaluable expression. A key difference between his functions and those typically dealt with is that he assumes that the arguments of the function are also seen in the final sequence.

There are several problems with Quick and Wilding's approach. The first is that they fail to pass the requirement for a general theory of music described above – namely, that the theory be able to integrate the multiplicity of properties experienced simultaneously in a piece of music. Of course, there's nothing wrong with a theory which is not a general "theory of everything," but explains one facet of a problem. More troubling is these authors' assumptions that music is context-free. This is blatantly false. One frequently sees progressions of the type $I^nV^nI^n$, which are not representable in terms of context-free grammars. These grammars don't cover cross-serial dependencies such as C D V/ C V/D, which is also common. Harmonic acceleration and deceleration is a reality that is not represented by context-free grammars. Yet despite these flaws, it remains true that the music these authors' systems produce sounds much more appealing than randomly chosen notes, indicating that context free grammars may at least be a simplification of some real grammatical process (one can here my slightly elaborated implementation of Quick's proposal here: https://soundcloud.com/user-332259139/context-free-grammar-music). Furthermore, the idea that music can be described using different harmonic layers is in line with the widely used Schenkerian method of analysis, which examines the "background," "middleground," and "foreground" of a piece.[25]

Another approach to musical grammars not based on much music theory is the used of

[25] Schenkerian Analysis in its strongest form (that every piece of music consists of a linearly descending background melodic line) is widely debated and not particularly convincing to this author, but his general idea of examining foreground, middleground and background is widely agreed to be useful.

Lindenmayer systems for generating musical surfaces.[26] A Lindenmayer system, or L-system, consists of not non-terminal and terminal symbols like most context-free grammars, but rather a single alphabet. Each symbol is on the left side of exactly one rule and on the right side an arbitrary amount of times in up to all of the rules, and rules are applied a given or arbitrary number of times starting with an arbitrary start symbol (an element of the alphabet). In musical applications, the symbols are typically notes (A,B,C). Thus an example

$$A \rightarrow ABAC$$

$$B \rightarrow BCC$$

$$C \rightarrow ACA$$

Start symbol: A

applied 3 times would create the sequence

$$A \rightarrow ABAC \rightarrow ABACBCCABACACA \rightarrow$$

$$ABACBCCABACACABCCACAACAABACBCCABACACAABACACAABAC$$

This method can generate some fairly interesting short melodies, especially when combined with a bit of randomness in the rules or rhythm (an implementation by the author can be heard here: https://soundcloud.com/user-332259139/lindemayer-system-music). However, it is not particularly promising as a theory of music, as the vast majority of music is not of this form, and as pursuing this approach for even a moderately-sized piece of music quickly becomes monotonous.

---

26 Manousakis, Stelios. (2006). *Musical L-Systems* (Master's Thesis). Retrieved from http://carlosreynoso.com.ar/archivos/manousakis.pdf

Music as Points in State Space

A common approach to generating music is through genetic algorithms.272829 The results are typically fairly poor, and it is easy to see why: the "fitness" functions used often simply specify qualities such as that one should not have too many leaps, or that the rhythm should be fairly regular. Such requirements are not even close to what is needed to define music. They fail the basic requirement of music generating algorithms to consider multiple dimensions of music at once. Such requirements also often will not generate anything that is "syntactic" at all, and, while I have not and cannot provide a precise definition of what syntax is in music, the reader should now be convinced that hierarchical structure must in some way be perceived in most music for it to be considered music. It is also worth noting that these genetic algorithms basically offer a very computationally expensive way of doing something that could be implemented directly with not much more than a Markov model – there's no reason one can't impose a restriction on the number of leaps probabilistically and/or with hard rules.

Despite its flaws, genetic programming with respect to music composition is worth considering for several reasons. As a model of how human composers actually compose, it is certainly not perfect, but it may have some value: humans don't mutate their products thousands of times until they reach a given fitness level, but they do start with a base product, and tweak it until it reaches a level of satisfaction (which presumably does involve, among many other factors, the extremely basic

---

27 Biles, John. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. *Proceedings of the 1994 International Computer Music Conference*.

28 Matic, Dragan. (2010). A Genetic Algorithm for Composing Music. *Yugoslav Journal of Operations Research,* Vol 20, pp. 157-177.

29 Jeong, Jae. (2015). Automatic Evolutionary Music Composition Based on Multi-objective Genetic Algorithm. *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems.* Vol 2.

requirements of music that such programs stipulate). However, I believe that the most significant contribution of genetic programming is the view in which composing music involves traveling through a search space, with some elements in the search space more promising than others. Music fundamentally involves a combinatorially large number of possibilities, which can be explored systematically as if one is traveling through the search space and tested for musicality.

A search space in which the sole "object" being represented is the musical notes, however, is hardly enough to work with. To determine a group of sounds' musicality, one needs to look at the multiplicity of properties and inner structures that the sounds have. One could argue that one could represent music as notes but simply apply functions that determine the music's properties at the time when one is judging the music's musicality. However, for the purpose of doing any serious analysis (and, therefore, when one does any serious generation task), it is useful to explicitly store the representations of music that these functions produce. This allows for comparing the properties of two snippets of music, or for looking at the distribution of musical properties. Thus, the representations of music in search space should consist of a large number of properties.

What would such a representation look like? As stated above, at any given moment, music has many features that exist all at once – rhythmic features, harmonic features, textural features, and melodic features. Each of these categories of features can be broken down into tens of different features as well, (see the table below).

Rhythmic Features	**Harmonic Features**	**Melodic Features**

• is_symmetrical (boolean) • longest_duration (float) • shortest_duration (float) • contains n_tuplets (bool) • n_tuplets (int) • durations (list of floats) • is_syncopated (bool) • is_augmented (bool) • is_diminished (bool) • length (float) • significant rhythmic patterns (list of list of floats)	• is_diatonic (bool) • is_triadic (bool) • mode_intervals (list of ints) • tonic_pc (int) • has_secondary_tonic_pc (bool) • secondary_tonic_pc (int) • min_dissonance (int) • max_dissonance (int) • chord_set_classes (list of list of ints) • chord interval vectors (list of list of ints) • has_tritone (boolean) • is_tonal_cadence (boolean)	• horizontal_degree_intervals (list of int) • vertical_degree_intervals (list of int) • pcs (list of int) • degrees (list of int) • has_leaps (bool) • number_of_leaps (int) • is_symmetrical (bool) • has_turn (bool) • has_arpeggio (bool) • is_stepwise (bool) • emphasized_pcs (list of int) • contour (list of int)

One potential approach to musical semantics that one could propose is one in which one considers all the possible values of each of the musical properties described above, along with whichever others one wishes to consider. There are a combinatorially huge number of such properties, but not infinite – assuming one is willing to make the reasonable assumption that the music itself is not infinitely long. Thus, the possible values of the musical properties as a whole define a finite space of sorts. I will call this space a "musical feature vector space" (in spite of the fact that the multiplicity of types involved make it not an actual vector space) or "musical feature space", and will refer to a given musical surface's values as a "musical vector." I will refer to any algorithm that can take a musical surface and produce its musical vector as a *music2vec* algorithm.

It is clear that such an approach can be useful for determining the semantics of music – especially

in comparison to conceptualizations of Mikolov's *word2vec* algorithm.30 The word2vec algorithm relies on associational semantics (the idea that a word's meaning is defined by the company it keeps). Using a large corpus of sentences, a neural network can be trained to produce vector representations of every word fed to it, such that words which are "similar" (seen in similar contexts) have a small Euclidean or cosine distance from each other. *Word2vec* is obviously not entirely comparable to *music2vec*. In *word2vec*, the actual vector representation is arbitrary and not interpretable by humans, and the only thing that matters is a word's position relative to other words. In contrast, *music2vec* creates representations of music which are completely interpretable to humans, and which are useful even when one only considers a snippet of music by itself. However, both assign semantic meaning based on an item's position in a space in which distances can be measured along multiple dimensions.

Even more usefully, *music2vec* can be compared to current research in "skip thought vectors," an extension of *word2vec*.31 Like *word2vec,* skip thought vectors take a large corpus of data, and assign vector representations to each word such that similar words are near in the n-dimensional space. However, the researchers behind skip thought vectors went farther, and studied the movement between words in a sentence in terms of this vector space. They then created a single vector for each sentence which embodied both the sum representations of each word in the sentence, and the movement between them. Similarly, one can talk about a larger piece of music in terms of the sum of the properties of its parts, as well as the type of movement in musical feature-space between the parts it is made up of.

Clearly, if one could produce such a representation, one could argue persuasively that one has knowledge of musical semantics. However, I'd argue that in addition, musical feature-space gives one

---

30 Mikolov, Tomas et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in neural information processing systems*.

31 Kiros, Ryan et al. (2015). *Skip-Thought Vectors.* Retrieved from https://arxiv.org/abs/1506.06726.

knowledge about musical syntax. One can say that musical semantics describes an individual point on feature-space. On the other hand, musical syntax describes a type of walk across feature-space. Such a walk can be described in terms of what dimension is changing (i.e., rhythmic, melodic, harmonic, or a more refined type of movement such as in largest duration), and in what the movement looks like (random, circular, linear, etc.).

This paradigm of "syntax = type of movement" may not sound like it fits well with the methods described above of reducing the concept of musical syntax to formal grammars describing harmony. However, in fact the new conception of syntax can be shown to contain Donya Quick's formulation of musical syntax. As discussed above, Donya Quick's grammar has rules such as the following:

$$I_{1.0} \rightarrow I_{0.25}\, IV_{0.25}\, V_{0.25}\, I_{0.25}$$

, where the numbers denote the relative length of each section. This rule may look similar to the well known $S \rightarrow NP\ VP$ of linguistics, but in fact they are different in a crucial way – the difference in *I* and *IV* is quantitative, entirely relative, and describes the relationship between the two sections rather than their absolute qualities. Thus, such a rule states the type of movement between sections as a one-dimensional vector of motion in a space of other possible musical features, and so describes one dimension of a circular walk around feature-space.

Example: Analyzing the Feature-Space of Mozart's Piano Sonatas

As a toy implementation of musical analysis based on feature spaces, 69 movements of piano sonatas by Mozart were processed and analyzed. The pieces were taken from the online database *kunstderfuge*, a source of .midi and .krn representations of thousands of classical pieces.32 Using the

32 See www.kunstderfuge.com

python library music21, the files were imported as individual parts, each consisting of lists of notes together with onsets.[33]

In order to produce the feature-space representation, music had to be segmented into individual parts. This was done on two levels – at the level of the phrase, and at the level of the measure. The reason behind this was twofold. There are types of analysis where it makes sense to compare very small pieces of music, and there are levels of analysis where it is appropriate to say something about a larger piece of music. Furthermore, as described above, there are certain properties of any small bit of music which is tonal (i.e., the category of music written by Mozart) which is relative to the scale being used. For instance, the degree of a note is defined by its position in the current scale. While measures are interesting to study on their own, they do not provide enough tonal information to determine what scale they are in. Fortunately, phrases in 18th-century music tend to have a uniform scale (with the exception of one or two modulations, or change of scale, per piece – and even those tend to be within "close" keys). Therefore, by determining the phrase structure, it is actually possible to perform more interesting analyses of individual measures.

The segmentation of pieces into phrases and meaningful measures is surprisingly complex. While measures can be defined by bar lines in written music (which are not provided in the digital representation), in fact for the sake of analysis one probably wants up-beats (notes which start before the bar line but are part of the following piece structurally) to be included in the following measure. Phrases are even more ambiguous: although empirically it is easy for even most non-musically-oriented listeners to segment phrases in Mozart's and comparable music, there are no clear rules for when a phrase ends and a new phrase begins.

---

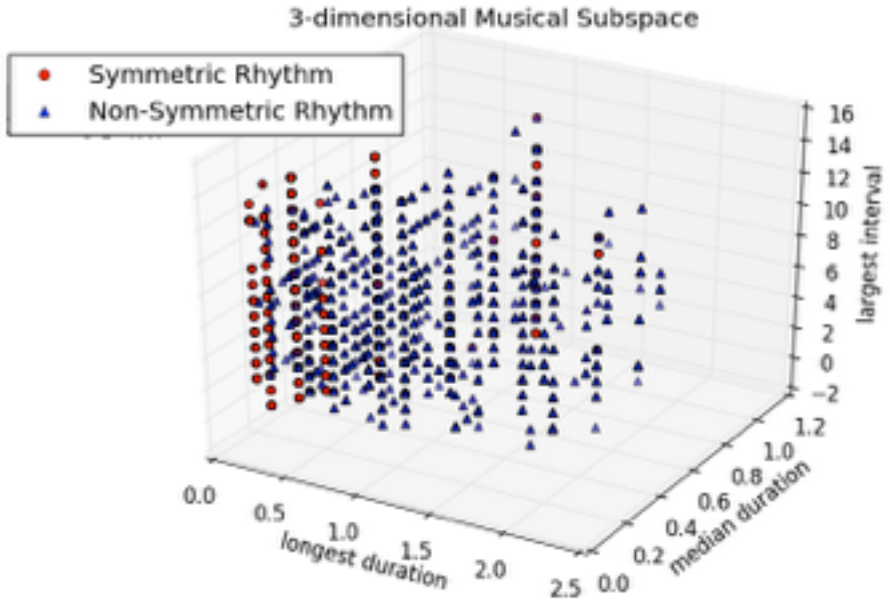
33 Cuthbert, Scott, and Ariza, Cristopher. (2010). music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. *International Society for Music Information Retrieval,* pp 637-642.

The algorithms used in this study were based on dynamic programming. Each segmented measure or part is given a "goodness" value, determined by its length (between 2 and 5 beats for the measure task and between 2 and 6 measures for the phrase task is preferred) as well as whether it ends on a rest, a long note, or a short note (long notes and rests are preferred, short notes are given a low score). The best segmentation is the segmentation that results in the best sum of the scores of each segment.

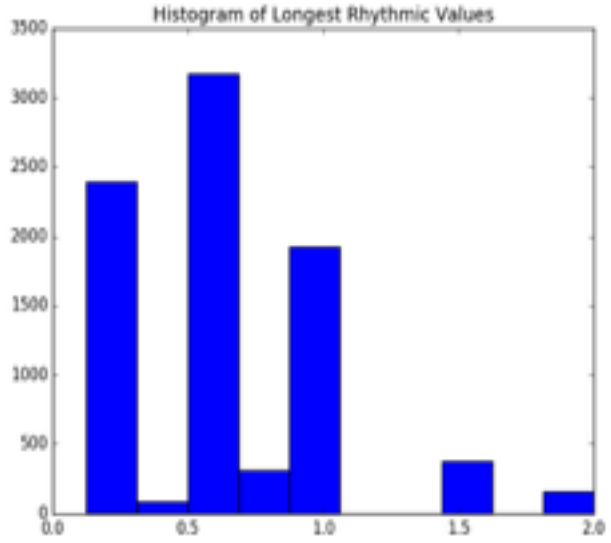
Another step that had to be taken in analysis was to determine the scales being used. Each phrase of each piece was assigned a scale. This was done using hill-climbing search: the best overall phrase-scale mapping was low in total scales, but minimized notes not in the scale. In addition, the reward was higher when the scale chosen would imply the existence of many tonic notes (notes whose pitch class is equal to the scale itself), as it is well known that in tonal music the tonic note is highly prevalent.

Results

The analysis of Mozart's piano sonatas allowed for interesting insights into Mozartian state-space. The distribution of certain features in the overall corpus was explored:

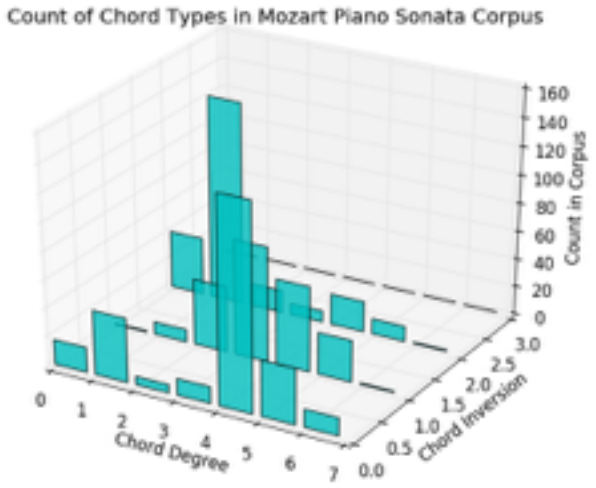


The above graph shows four dimensions of musical subspace described by various points across the Mozartian corpus. Certain correlations become apparent: for instance, symmetric rhythms most often



involved only small duration values. The most common longest duration values were powers of two (0.25, 0.5, and 1).

Similarly, one could explore the distribution of chords in the musical space:



Unsurprisingly, the most common types of chord are the tonic and dominant. It is also unsurprising that certain chords are seen more in first inversion or in second inversion (the inversion being the way in which the notes are ordered vertically).

One result that would surprise no musicians involved movement in state-space:

In the above graph, each color represents a different sonata. Each sonata's mapping of phrases to

scales was taken, and normalized so that the first scale was labeled as 0. The normalized lists were then plotted as function of the phrase number. It is clear from the graph that every piece has a slightly different structure with regard to the movement around state-space in the scale dimension over the course of the piece. However, there are also some striking regularities. Every piece ends in the key in which it starts. Most pieces modulate (change keys to) either the perfect fourth, perfect fifth, or minor sixth. Some pieces switch back and forth between keys several times, while most move away from the home key only once. This movement pattern is part of what defines a sonata; I would argue it is part of a sonata's "syntax."

It is important to note that the analyses done on the Mozart corpus would have generated

meaningless data if it were done on a corpus of Schoenberg's atonal works. On the other hand, if a set-theoretic analysis (of the sort that would explain a Schoenberg piece) was done on Mozart's work, all that would be discovered is that there are many triads; the system of tonality which drives the piece is not well-explained by set theory alone. Certain pieces lend themselves to certain types of analyses. Thus, the state-space one explores must be equipped to handle the type of music one is interested in.

Limitations to the State-Space Approach

At first, treating music as a pattern of movement across state-space seems to solve all of our problems. It is a general method that can be tailored to work with any type of music: If you're interested in Hindustani music, describe the *tala's* properties as points in the space; if you're studying Messiaen, describe the limited modes of transposition used. It also provides a neat definition for musical meaning and structure; syntax is movement in space and semantics is the point in space itself. However, there are several issues that make using state-space to generate music somewhat inviable.

The first issue, which is of concern with regards to both the analysis and generation of music, is that there is no clear length l such that a snippet of music of length l should be described as a point in feature-space, and a snippet of music of length $> l$ should be described in terms of movement along feature-space. Some very short pieces of music (<4 bars) can be described as a tiny motif and its inversion (which presumably should be annotated as a type of transformation of a point along feature-space, or a walk along feature-space), while some fairly long pieces of music are best understood as a whole "tune." In addition, because of the hierarchical nature of music, one may resort to different "levels" of movement. For instance, one might describe the movement between a motif and its

inversion, and then a transposition of that movement a tritone away (moving both points by a constant factor in one dimension). This may seem like a trivial concern, but it actually gets at a fundamental problem in our search for musical syntax and semantics – the line between them is much more blurred than in natural language.

Another issue is the inability to describe transformations within state-space. Music often involves transformation of material. Sometimes this can be easily described as linear movement within state-space, but often the transformation is more complex (e.g., multiply every duration by 1.5). This problem is perhaps less pressing in that a) many transformations used by modern composers are not actually heard, and thus could be considered less crucial and b) in some cases this issue can be partially resolved by computing some "normal form" as a dimension of the musical space (in the case of augmenting and diminishing rhythm, this would involve storing the rhythm normalized to have a beginning duration of 1.0, as well as the longest duration). However, as modern music has developed, composers have begun using more and more advanced transformations. These transformations are part of the musical object, and therefore should be first-class elements of whatever analysis is attempted.

A final issue, pertaining only to the creation and not the analysis of music, is that there are areas of musical space that cannot be entered due to logical contradictions. For instance, one cannot have a "whole-tone" scale and have major chords, as the whole-tone scale does not allow for the development of such chords. Similarly, one cannot have a complete 7^{th} chord in a 5-note rhythm with only movement by major seconds. Thus, one cannot simply choose a value for each dimension of music and create a piece of music with those values.

One way that one could perhaps get past this final issue is to prioritize musical features. One successful way of doing this will be integrated into the next section; however, it is worth mentioning

one way this CANNOT be done. One could imagine creating logical variables that describe musical features, and logical clauses that embody some musical fact (the concept of symmetrical rhythm could be encoded as (IS_SYMMETRY and [at least one of each conjunction of every type of duration encoding such that the durations the variables encoded would be symmetrical]). One could presumably come up with a desired position in state-space, assign logical variables according to that position; and run a SAT-Solver preloaded with those variables to see if it is possible; if it is, return the actual music encoded in the result, and if it's not, remove some of the dimension assignments and run the SAT-Solver again to see if it works.

Why is the logical approach not possible? It turns out that there is no obvious way to encode music using logical variables. One could try note_*x*_has_pitch_*y* for all pitches and note indices, and note_*x*_is_dur_*y* for all durations and note indices. Assuming 60 pitches, 30, durations, and 20 notes, this results in 1800 variables (a large number but not impossible) of the surface music alone. However, the number of clauses is combinatorially larger. Searching for a symmetrical 8-beat rhythm using a SAT-Solver would be incredibly difficult; there are many possibilities that are symmetrical, as well as many, many more which aren't.

Music as the Evaluation of a Program

Thinking about musical state-space clearly advanced our understanding of music: It is the first idea proposed that really takes advantage of music's incredibly high dimensionality. At the same time, imagining a flat space doesn't get at how music is put together; it explains neither the logic of what musical attributes are possible, nor what functions can be used to get there. Musical state-space is a passive description; we need an explanation involving actions. One way to do that is to pivot slightly

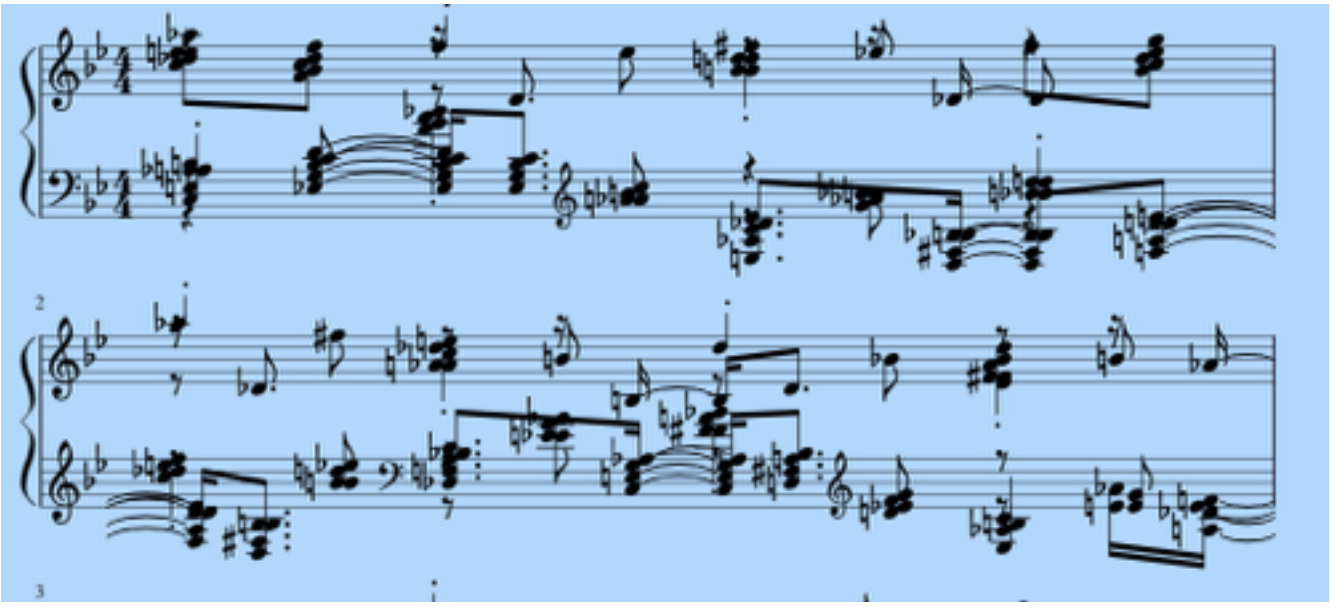
from the study of musical features to the study of musical objects. I define a musical object to be any facet of a piece of music. This obviously includes the dimensions measured in feature-space, but crucially also includes musical functions.

Fortunately, computer scientists have already come up with a way to talk about objects that include data and functions alike – namely, type systems. Type systems allow for discussing what possible types of objects there are in a universe; in particular, we want to know what types of objects exist in music. It turns out that there is a very rich array of types of musical objects. For instance, one can talk about the type of rhythms, which is a list of floating-point numbers. Alternatively, one can talk about the type of the function *transpose*, which takes a interval and a melody as parameters and returns a new melody. One can even imagine a higher-order musical function which takes another musical function mapping notes to chords and applies it to every nth note. The types can themselves be objects: for instance, a function might take the name of a musical parameter (rhythm, harmony, etc.) and a melody and apply a transformation related to that parameter to the melody. One can have inheritance patterns, where an A can be a B without a B necessarily being an A; any horizontal interval is an interval, but not every interval is a horizontal interval.

Like state-space theories, musical type theory is important in that it can be generalized across musical cultures in a way that musical grammars cannot. There are different musical objects in different cultures – there is no Western equivalent to the raga, or to the Bembe wheel. However, I hypothesize that the different musical objects in musical cultures all share the property that they can be described in terms of decimal or integral numbers, categorical or boolean variables, functions, or combinations of these types created through function composition and list application.

Unlike state-space theories, type theories lend themselves to the creation of new music. Just like

programs can be composed of functions and variables of various types, a piece of music can be composed of a system of interacting objects. Consider the example below, which can be heard here: https://soundcloud.com/user-332259139/messiaen-like-generated-music. (Warning: while I have played this to musicians who believe it is much better than the previous things I composed, it is very modern sounding and may be perceived as patternless banging on a keyboard).



This music was composed by bringing together several musical "objects," namely

- the set classes [01348], [01458], [02358], and [01358]
- the variable is_symmetrical = True
- the subset function
- four horizontal interval variables
- the invertSetClassFunction
- the parallel_sequence function, with parameter relative_duration = 1.5

These objects were brought together in this fashion: 4 symmetrical rhythms were generated. Each rhythm was transformed into a soprano melody by taking the chosen horizontal intervals, and

making the first element of each rhythm the full set class, which each following element corresponding to a subset of the set class. The bass was created by inverting every set class of the soprano. The result was sequenced so the same rhythm played 3 times in the soprano, at the same time as it played twice in the bass.

It is worth noting that we do not know every parameter of the music. We did not determine the median duration, or the overall distribution of major or minor keys. While this is partly due to the fact that it is a simple example where more than usual is left to chance, it is also generally true of music as it is normally composed that the composer chooses which parameters to worry about in a given time. Some parameters will always be salient, like if the music gives an overall impression of being fast or slow, tonal or atonal. But every composition process will necessarily neglect some of the "features" that would be in a state-space. As mentioned below, Mozart did not pay attention to the set classes in his piece or attempt to organize them in any manner. Although he surely knew that set classes were heard, their organization was not part of his process or system.

Types and Program Synthesis in Music

As mentioned above, the most exciting aspect of musical types is the way they can compose (as in function composition) into actual musical pieces. The above example was generated by writing a program that carried out the processes described. However, we can actually do better in our quest to find uses of this particular conception of musical meaning with respect to musical generation. Instead of specifying what each program should do (and thus what musical objects interact with each other and how), we can automate the process of generating music-producing programs.

The proposed program synthesis paradigm for generating music works as follows – a non-deterministic program generation algorithm generates a program, which in turn generates a piece of music (formatted as a wav file). Because the generated programs are generally non-deterministic, one can say that the generated program generates a family of related compositions.

As discussed above, this method of automatic composition through program synthesis is modular, and can encompass specific theories for music. A function can be developed which generates a chord progression using context-free harmonic grammars, and can be employed in the program. One can create a pc-set type, and define functions that operate on such pc-sets. The meta-theory is fully extensible to any desired additional musical types and transformations.

While I don't believe that anyone has explored music as programs in depth before, I am borrowing from Heinrich Taube's idea that he articulated in Notes from the Metalevel, in which he explained that programs can serve as meta-scores for given pieces of music.[34]

An Overview of Program Synthesis

What is referred to as "program synthesis" varies wildly depending on the application. In all cases, it involves the generation of working code in some language based on a given input. However,

---

[34] Taube, Heinrich. (2004). *Notes from the Metalevel: Introduction to Algorithmic Composition.* Netherlands, Swets & Zeitlinger Lisse.

that input can range from an "oracle" that solves the problem given (although presumably not as efficiently as the generated program would), to logical specifications detailing some desired property of the output, to example input and output pairs, to natural language input.[35] In addition, the desired output may be specific to a DSL, may involve only certain efficient computations such as bit-vector manipulation, may have certain control-structure properties such as being loop-free, may be required to have a certain syntax, etc.[36]

Program synthesis developed originally out of the development of interactive theorem provers coupled with the realization that there is a tight link between proofs and programs (the Curry-Howard isomorphism).[37] Thus, original program synthesis techniques typically involved extracting functional programs from theorem provers. Alternatively, other algorithms do a random search of program space coupled with an SMT solver or some other verifying program.[38][39] These types of algorithms have improved as SMT solvers and interactive theorem provers have improved, and as tools for search have improved.[40] However, as the search space is exponential, they still have not been able to be applied to problems of any realistic scope.

Recently (since 2015), work has been done in generating programs in typed, functional languages based on type theory. Osera and Zdancewic present a method, which Osera expanded on in

---

[35] Sumit Gulwani. (2010). Dimensions in Program Synthesis. *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming,* pp 13-14.

[36] Rajeev Alur et al. (2013). Syntax Guided Synthesis. *FMCAD,* pp. 1-17.

[37] Christoph Kreitz. (1998). Program Synthesis. *Automated Deduction – A Basis for Applications*, pp. 105–134 .

[38] C Mohring, B Werner. (1993). Synthesis of ML Programs in the System Coq. *Journal of Symbolic Computation* Vol 15-5-6.

[39] See https://emina.github.io/rosette/

[40] Lim, J., & Yoo, S. (2016). Field Report: Applying Monte Carlo Tree Search for Program Synthesis. *Search-Based Software Engineering*. Springer International Publishing.

his thesis, in which the input is a given function name, its type, any auxiliary functions it can use, and examples of what it should output; the output is a completed function.[41] The program systematically creates "refinement trees" out of the examples, branching when it is clear from the examples that a single case will not suffice. This method also takes advantage of the grammatical nature of correct program code.

Synthesis of Music in the Context of Program Synthesis at Large

The problem I will be discussing is somewhat easier than the tasks undertaken in previous program synthesis research. Generating music does not require searching an exponentially large search space for the one "right" answer; when done right, the programs generated should each come up with different, equally musical results. Unlike many program synthesis researchers, I am not starting from the ground up with no functions available, but rather am assuming the existence of useful musical functions. In addition, I am severely restricting the types of statement available in the programs to assignment of variables to either a value, a list of values, or an expression with a pre-defined function as its head and previously assigned variables as the function's parameters. This is actually in line with prior program synthesis research which also severely restricted the types of statements that could be expressed.42

The Language of Generated Programs

The language Python was chosen for representing the generated functions. Python is an

---

41 Osera and Zdancewic. (2016). Type- and Example-Driven Program Synthesis. *Symposium on Trends in Functional Programming.*

42 See Alur et al, 2013.

imperative programming language which treats functions as first-class citizens. Thus, it was possible to incorporate features such as loops and side effects into functions, while it was also possible to pass functions as arguments. Python is dynamically typed, but supports adding static fields to functions, which provides a useful interface for optional typing. The addition of static fields to functions also makes it easy to classify functions, and to encode semantic information about each function.

With metaprogramming through metaclasses and decorators, Python is a very extensible language. Two decorators were developed that were used on most functions accessible to the generating program:

Function Currying without Order

Most functional programming languages support currying – passing an argument to a higher-order function and returning a new function of arity one less than the original function. Python doesn't natively support currying. However, it was easy to produce using decorators. Crucially, the type of currying I introduced does not have an order of the different arguments, but instead relies on keywords to obtain the correct semantics.

```
def f(a,b):
     return a + b
f(a = 2) //evaluates to a function
f(a = 2)(b = 3) //evaluates to 5
f(b = 3)(a = 2) //evaluates to 5
f(a = 2, b = 3) //evaluates to 5
```

This is useful in contexts of music. For instance, if one is considering a function *transpose* which takes

an initial melody and an interval to transpose by, one might want to vary the melody being transposed, or one might want to vary the transposing interval – there's no natural order. Currying through keywords solves this problem.

Mapping

As discussed above, music typically has hierarchical structure. One way of describing such hierarchical structure is in the use of ordered lists. A common structure in music is a period, or two phrases both made of two parts. This could be described as [[a,b],[a,b]] (this isn't perfect, as there's no way of knowing merely from the structure that the two a's are similar, but describes the tree structure fairly well). Thinking about nested lists makes the most sense when imagining certain movements in musical feature-space. Consider the example above, of a motif and its inversion which are then transposed first a tritone and then an octave away. This can be viewed as moving in inversion-space to produce an inner list, and then moving in transposition-space to produce three copies of this inner-list with different points in the dimension of "first-pitch." This would produce a structure [[a_{11},a_{21}], [a_{12},a_{22}], [a_{13},a_{23}]].

How can one create such structures algorithmically? Define a function `map1` which takes arguments that are either non-lists, lists, lists of lists... etc. of values, and returns a hierarchical ordering defined by those lists. For instance, assume one calls `map1(rhythm, pc, start_note)` where the `rhythm` has one value *a*, the `pc` *i*s a length-two list *[b1,b2]*, and the `start_note` is a length-three array [*c1,c2,c3*]. This would be evaluated as follows:

```
val_dict =
```

```
[
      [
            {'rhythm':a,'interval_direction':b1, 'start_note':c1},
            {'rhythm':a,'interval_direction':b1, 'start_note':c2}, ]
            {'rhythm':a,'interval_direction':b1, 'start_note':c3}, ]
      ]
      [
            {'rhythm':a,'interval_direction':b2, 'start_note':c1},
            {'rhythm':a,'interval_direction':b2, 'start_note':c2}, ]
            {'rhythm':a,'interval_direction':b2, 'start_note':c3}, ]
      ]

]
```

This can be generalized to arbitrarily nested lists. One can also define a similar function `map2` which only differs in that two lists with the same size are not nested hierarchically, but rather the corresponding elements are grouped pairwise.

Then define a decorator `map_dec` which takes an arbitrarily nested list of dictionaries of parameter-value pairs, and returns a mapped list that is the result of evaluating f with respect to the parameters in each dictionary.

```
map_dec(f)(val_dict) =
[
```

```
    [
        f(a, b1, c1), f(a, b1, c2), f(a, b1, c3)
    ]
    [
        f(a, b2, c1), f(a, b2, c2), f(a, b2, c3)
    ]
]
```

One thing to note is that there are cases where the function `f` should actually take a list (for instance, when one of the arguments is a chord). Thus, for this to work, it must be known in advance what types the arguments should have. In practice, this is not a problem most of the time, as the only functions which I have applied the mapping decorator to are the repertoire-specific functions, which take specific musical objects as functions. Given the limited usage and scope of the framework, one can assume that all arguments of a given name (such as "chord" or "interval") are of the same type. Therefore, one can extract from the function the name of the arguments, and check each argument against a list of argument types to figure out proper mapping.

Types of Functions and Values Used in the Generated Programs

As discussed above, there are known musical types which are given and returned as arguments to the various functions. Some of these types have a few, primitive values – for instance, a pc has a value of 0 through 11, an n_tuplet has a value in [3,5,7,9,10,11], etc. The generated programs have access to the full range of these values. Other values are too great in number to be listed in a

dictionary, and are only returned as the outputs of functions.

There are two types of functions used in the generated programs – what I refer to as “repertoire-specific” functions, and “functional” functions. Repertoire-specific functions include functions that operate on musical objects according to a given specific musical theory, such as set theory or tonal theory. Repertoire-specific functions have known input-types and output-types. Examples include the following:

```
messiaenModeKeyToScale(messiaen_mode, key)

genIntervalVector(num_notes)

keyModeChromaticismToTonalProg(key, mode, chromaticism)
```

These examples come from three different “theories” of music (Messiaen's personal theory of modes of limited transposition, set theory, and tonal theory), which shows how any repertoire's theories can be incorporated. A subset of the repertoire-specific functions and primitive values are described in the appendix.

In addition to repertoire-specific functions, there are functional functions, which are polymorphic and take either repertoire specific functions or musical values as arguments. Examples include:

`repeatN(n, x)`, which repeats `x` `n` times. If `n` is the value 4 and `x` is the value “inexact”, the resulting value would be `[“inexact”, “inexact”, “inexact”, “inexact”]`. It's important to note that this function also has the `mapping` decorator, so if `n` or `x` is an array, the resulting value will also be a two-dimensional or larger-dimensional array.

`execN(n, f)`, which executes the function `f` `n` times.

`repeatApply(x_0, n, f)`, which results in the array `[x_0, f(x_0), f(f(x_0))...]` of length `n`.

`applyTransforms(x_0, n, fs)`, which returns an array whose first element is `x_0` and whose subsequent elements are each of the functions in `fs` applied to `x_0`.

Functional functions have the `build` field. This field consists of a function which takes two arguments, `type_` and `shape`, and returns the required arguments that would need to be inputted in order to achieve the given shape of the array that contains at its most nested level objects of the given type.

The writeScore function is called at the end of every generated program. It takes an arbitrarily nested list of Melody objects, and renders it to a CSound (sound synthesis library) file, which is immediately processed to produce a wav file.

Putting It All Together

Given the above setup, generating the programs becomes a graph generation problem. The final output one wants is a 'melody' type object of a given dimension *x*. One creates a melody object of dimension *x* in one of several ways. If one has already generated a melody of dimension *y*, one can repeat a subset of that melody that has dimension *x*. Alternatively, one can find a function whose output is a melody, and instantiate each of the arguments with the right dimensions $x_1...x_n$ so that the

final output has dimension *x*. The melody object can be modeled as a node in a graph, where the name of the function used to generate the melody object is a propery of the node, and the arguments that are passed to that function are nodes which are pointed to by edges from the melody object. Then the arguments are recursively expanded in the same way that the melody was. The expansion is not infinite, because some objects are either generated solely from functions which take no arguments, or are given a small number of primitive values rather than being generated by a function.

As mentioned above, the functions used in the program may be repertoire-specific, or they may be functional. Repertoire-specific functions can be divided into transformations (functions that take a melody and return a different melody) and generative functions (functions that take certain objects and return a different type of object). Transformation functions are constrained to only be used when both the original object and the object which is transformed appear in the final output; otherwise, the program could consist of tens of transformations of the same melody such that the original melody isn't really perceived at all. Functional functions often take functions as arguments, which is not a problem due to the use of the currying decorator.

From the graph, it is easy to generate the final program. The nodes of the graph are topologically sorted, such that the final output "melody" which all the other nodes help create is last. Nodes are equated with variables. Each node is assigned a name according to its type and position in the program, which is referenced by the nodes further down in the program. The function `writeScore()` is called on the final variable, which is of course a melody.

Results

Below are two example programs:

Example 1 (can be heard here: https://soundcloud.com/user-332259139/program-synthesis-music):

```
contour0 = genContour()
num_note_range0 = [(1, 4)]
num_pat_type0 = ["every_other", "decreasing"]
num_notes0 = genNumNotes(map1([("num_pat_type", num_pat_type0),("num_note_range",
num_note_range0),]))
int_vec0 = genIntVec(map1([("num_notes", num_notes0),]))
contour1 = repeatN(n = 2, x = contour0)
sig_rhythm0 = [[0.5, 0.25, 0.25], [0.75, 0.25], [0.75, 0.25], [0.5, 0.25, 0.25]]
length0 = [6.0, 4.0]
rhythm0 = genSigRhythm(map1([("length", length0),("sig_rhythm", sig_rhythm0),]))
melody0 = cycleIntVecContourRhythmMelody(map1([("int_vec", int_vec0),("rhythm",
rhythm0),("contour", contour1),]))
writeScore(melody0, "t1")
```

This example involves a contour object, an object representing a pattern in the number of notes per chord, measure-length objects, interval vector objects, a repetition object, and some objects representing interesting rhythms.

Example 2 (can be heard here: https://soundcloud.com/user-332259139/program-synthesis-music-2):

```
sig_rhythm0 = [[0.25, 0.25, 0.5], [0.75, 0.25]]
scale_type0 = "whole_tone"
```

```
length0 = 4.0
rhythm0 = genSigRhythm(map1([("length", length0),("sig_rhythm", sig_rhythm0),]))
aug_or_dim0 = "augment"
exact_or_inexact0 = ['inexact', 'exact', 'exact']
rhythm_to_rhythm0 = augDim(map1([('exact_or_inexact',exact_or_inexact0),
('aug_or_dim',aug_or_dim0,)]) )
rhythm1 = applyInnerTransform(x_0=rhythm0, f=rhythm_to_rhythm0, type_ = "rhythm" )
mode0 = 0
start_octave0 = [5, 5]
key0 = 3
scale0 = keyModeScaleTypeToScale(map1([("scale_type", scale_type0),("key", key0),
("mode", mode0),]))
melody0 = startOctaveScaleRhythmToWanderMelody(map1([("start_octave",
start_octave0),("rhythm", rhythm1),("scale", scale0),]))
writeScore(melody0, "t2")
```

This example involves several musical objects, including a transformation (augmentation), whole-tone scales, and interesting rhythms.

Music as Composition of Types and Formal Semantics and Syntax

Music as composition of types is not only useful from a computer science perspective – it actually fits very well with linguistic approaches to semantics and syntax. In categorial models of formal semantics, typed lambda calculus is used to assign meanings to each word, which, when put together, yield a predicate calculus expression. For instance,

Kim walked and fed the dog.

Can be interpreted as follows:

Kim: k

walked: λx[Walked(x)]

and: λxλyλz[x(z) & y(z)]

fed: λxλy[Fed(y,x)]

the: λx[x]

dog: d

Kim walked and fed the dog: λxλyλz[x(z) & y(z)] (λx[Walked(x)]) (λxλy[Fed(y,x)] (d)) (k) ==

Walked(k) & Fed(k,d)

This predicate logic sentence can be understood either intensionally (as a function from time-worlds to truth values) or in terms of set theory (k is the set of individuals with the properties of Kim, Walked described the set of individuals who walked, etc.)

Music is not exactly equivalent, but like linguists compose a sentence's meaning out of a list of meaningful words using lambda calculus, in a simplified model of music generation we can compose aspects of music to come up with the music's overall presentation. This model doesn't map well to the intensional view of language, as music doesn't really have truth value, but it can be understood set theoretically.

Assume a universe in which all musical objects exist. The expression `λxContour(x)([1,3,2])` describes the set of things in the universe which are contours and have numerical value `[1,3,2]`. Similarly, the expression `λx[Start_pc(x)](0)` describes the set of starting pitch classes with numerical value 0. Now define a family of functions called `combine`, of different arities,

which take sets of musical objects and a different musical type and returns the set of musical objects that have that type but also have the properties of those objects:

```
λwλxλyλz[combine(w,x,y,z)]('melody') (λxContour(x)([1,3,2]))
(λy[Start_pc(y)](0)) (λz[rhythm(z)]([0.5,0.5,1.0]))
```

describes the set of all melodies with contour [1,3,2], starting pc 0, and rhythm [0.5,0.5,1.0]. The "meaning" of the set of music is defined by the musical objects that make it up, just as sentences are defined by the words that make them up. The isomorphism isn't perfect; words are heard consecutively, while aspects of music are heard simultaneously. However, we can also model musical functions and transformations that happen over time using lambda calculus:

```
λzλn([z, transpose(z,n)]) (
    λwλxλyλz[combine(w,x,y,z)]('melody') (λxContour(x)([1,3,2]))
(λy[Start_pc(y)](0)) (λz[rhythm(z)]([0.5,0.5,1.0]))
    ) (2)
```

describes the set of melodies which correspond to a melody described above followed by that melody transposed by 2 pitches.

One way in which musical lambda calculus may appear different than linguistic lambda calculus is that in linguistics, all elements are of some combinations of types *s* (world-time pair), *e* (entity), and *t* (truth

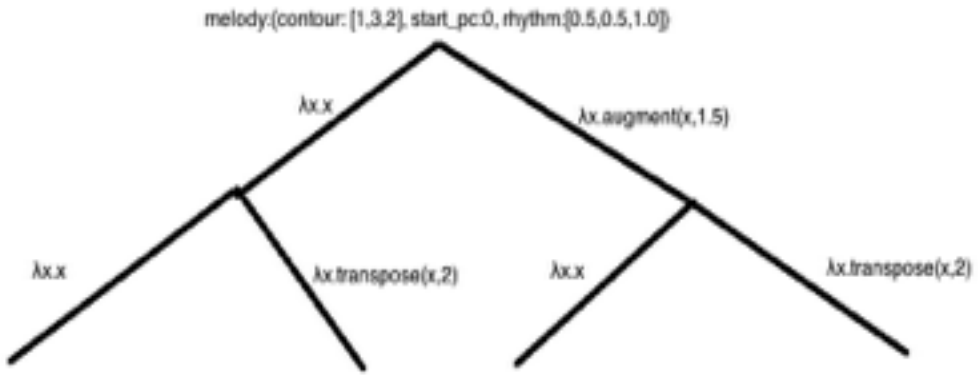
value). For instance, the word walked, described as the predicate λz[Walked(z)] would be taken to have a type <e,<s,t>>, where it takes an entity (presumably a person), and returns a function which takes a world-time pair and returns a truth value (whether that person walks in that world-time pair). On the other hand, I have defined many musical "types" such as contour, interval, duration, pc, pitch, chord, etc. However, as ultimately non-timbral music is composed of pitches and durations, it may be possible to create a "two-sorted" musical type system, like the two- or three-sorted linguistic type systems.

Music as programs also allows for traditional syntactic methods, specifically describing hierarchical tree structures. As mentioned, in my program synthesis and lambda calculus descriptions I use nested lists to describe hierarchical structure. Thus, the expression

```
λwλd([w, augment(w,d)]) (
[λzλn([z, transpose(z,n)]) (
    λwλxλyλz[combine(w,x,y,z)]('melody') (λxContour(x)([1,3,2]))
(λy[Start_pc(y)](0)) (λz[rhythm(z)]([0.5,0.5,1.0]))
    ) (2)]
) (1.5)
```

, which describes melodies with certain properties preserved followed by being transposed by 2, with the result being preserved and followed by the same objects augmented by 1.5, can be described hierarchically as follows:

This description of music as a tree is not perfect, because it does not highlight the similarity between



the two edges marked `λx.transpose(x,2)`. However, it does offer a way of segmenting a musical piece hierarchically based on its content that does not assume the only type of structure is harmonic structure. This interpretation supports the conclusion that type theoretical processes produce an explanation of music that is comprehensive, practical in terms of its ability to generate real music, and consistent with existing linguistic theories.

Fuzzy Logic, Fuzzy Membership, and Schema Theory

Logic and set theory are useful when partitioning music into different sets according to its properties. However, sometimes it is useful to talk about degrees of similarity between two musical objects, whether they are primitive objects (such as a pitch) or a combination of other musical objects. Such an approach in music has generally been relegated to Schemata Theory.43 According to certain formulations of musical schema theory, one can have a characteristic musical object, as well as objects which are similar enough to be identified with the characteristic object, but to a lesser degree. For

---

43 Chandler, Paul. (2005). *Music and Schema Theory.* Retrieved from http://tzone.org/~okko/html/documents/music_and_cognition.pdf

instance, a theme may often present with a jagged rhythm, ascending scales, and a crescendo enough to become the foundation of a schema; the same theme without a crescendo would "belong" to that schema, but not fully.

A natural formulation of these intuitions which takes into account the use of lambda calculus to describe musical objects involves fuzzy logic and fuzzy set theory. As discussed above, the phrase

```
λwλxλyλz[combine(w,x,y,z)]('melody') (λxContour(x)([1,3,2]))
(λy[Start_pc(y)](0)) (λz[rhythm(z)]([0.5,0.5,1.0]))
```

can be taken to refer to the set of melodies with contour [1,3,2], rhythm [0.5,0.5,1.0], and starting pitch class 0. But what happens when one takes a melody in this set, and applies the `make_jagged_rhythm` function? We already have ways of representing the fact that a function is being applied to an existing melody, but it's also worth noting that the nature of the transformation is such that the resulting melody, which has contour [1,3,2], starting pitch 0, and perhaps rhythm [0.75, 0.25, 1.0], seems like it almost fits in the starting set (only the rhythm is slightly changed, and even the rhythm is still recognizable). One could assign a degree of membership of the resulting melody to the described set. It would then be possible to categorize functions according to the extent to which they change the membership of the objects they are applied to in different sets.

Music and Complexity Theory

Another advantage of the "music as type composition" paradigm is that it makes it easier to discuss the complexity of music. Very little in the literature has been said about the relationship between music and complexity theory. However, one can easily make a case for a certain type of complexity theory's use in analyzing music. While it is certainly true that music is context-sensitive,

merely saying that it is beyond the complexity of context-free structures seems to say too little about music's complexity. There seems to be a huge gap between the structure $A^nB^nC^n$, and the complexity of a Mozart sonata. A more useful model of complexity for music may be Kolmogorov complexity. According to this definition, the complexity of a piece of music is the smallest set of rules which can completely replicate the piece.

This definition is useful in that it is somewhat possible to quantify differences between pieces of music's complexities, and to form hypotheses about the relationship between complexity and perception of the piece. For instance, Debussy's *De Pas Sur La Neige* is an example of a very compressible piece of music – following the rules of moving up and down on a melodic line while alternating two ostinati gives you a good approximation of the piece. On the other hand, the main theme of Dvorak's Eighth Symphony seems to be far less compressible, as there are no small sets of rules that would regenerate the tune. One problem with using Kolmogorov complexity as a method of describing music is that it is uncomputable. Unfortunately, no one has yet created a good method for approximating the Kolmogorov complexity of music. However, as discussed above, with the paradigm of program synthesis, one can generate music which has a certain description taking a given number of lines of code.

One issue to keep in mind when discussing the complexity of a program generating a piece of music is that complexity defined by lines of code is programming-language dependent. Generally, when one is discussing the complexity needed to implement something in a programming language, there is an assumption that no “libraries” and only “core features” are used; however, such an assumption doesn't need to be held (and what constitutes a library call vs a core feature is not always clear). This has real, musical consequences when one is talking about the complexity of a piece of

music. If one assumes that all of the rules of tonality have to be included in the description of a tonal piece of music, it will necessarily look very complex. If, however, one can assume the existence of some function that generates standard tonal progressions, the piece of music might appear extremely simple. Thus, the apparent complexity of a piece of music is necessarily dependent on what assumptions one makes about the "libraries" one can call about – or, in more musical terms, what conventions one starts with. This phenomenon is not only formal, but psychological: Listeners perceive of atonal music as more complex than tonal music until they are familiar with atonal set-theoretic operations. Listeners not familiar with Eastern rhythm perceive of it as more "complex" than Western rhythm, which I hypothesize is in part because they don't see the generators of Indian rhythms as within their musical vocabulary, or library of functions.

Other Things One Would Expect to Discuss When Talking About "Musical Meaning"

In this section, I will touch briefly on the computational/mathematical approaches to music that I do not believe are contenders for "grand theories of music." I will then discuss how to take advantage of the highly-constrained nature of tonal music. Finally, I will consider music where one has to consider additional types of meaning and structure, including film music, vocal music, and sonification.

The "Honorable Mention" Theories

I believe that the four paradigms described above – namely, music as probabilistic process, grammatical process, movement in state-space, and type-theoretical process, are the main contenders for a general theory of musical meaning and structure. However, there are several other ways of looking at music mathematically that have received a lot of attention, and it would be remiss not to discuss them.

Significant progress has been made in describing the relationship between combinatorics and music.[44][45] Combinatorics is useful for music if only in order to realize what problems can and can't be solved by certain methods. For instance, one might want to take a list of all possible measures containing 16 sixteenth notes, and then filter it to include only those with certain properties. However, even if one restricts the range of such a measure to an octave (12 notes), one still would have way too many possibilities (12^{16}) for such a task to be feasible. On the other hand, if one wants to find all possible realizations of an interval class [0134], one would find that there are only 24 such options – a

---

44 Read, R. (1997). Combinatorial Problems in the theory of music. *Discrete Math*, Vol 167-168, pp. 543-551.

45 Nolan, Catherine. (2000). On Musical Space and Combinatorics: Historical and Conceptual Perspectives in Music Theory. *Bridges – Mathematical Connections in Art, Music, and Science.*

number which makes most computations tractable. This may seem irrelevant, but in fact much modern music is built by combining different permutations or realizations of the same principal structure. For instance, serial music is written by constantly permuting and transforming as 12-tone row, which combinatorics tells us has 12! realizations. Oliver Messiaen systematically attempted to reduce the number of permutations of a structure, for instance by applying the same transformation repeatedly until the original permutation is reached, such as in the sequence "1 2 3 4 5" → "1 3 5 2 4" → "1 5 4 3 2" → "1 4 2 3 5" → "1 2 3 4 5".46 Unfortunately, the awe-inspiring nature of musical combinatorics (that there are a virtual infinity of possibilities in most situations) means that unless one is deliberately going about writing a piece based on combinatorics (as Messiaen did), knowing permutations and power sets often does no practical good.

Musical set theory has been established as one of the two leading alternative systems to tonality in post-modern music (the most common besides serialism).47 Musical set theory describes various musical structures in terms of sets. The most simple example of a musical set is a pc-set such as [1-2-10], which suggests that the set consists of pitch classes 1, 2, and 10 (pitch-classes are numerical names for pitches such as A,Bb,C#, etc.). However, many different structures can be described as sets, from interval classes to contour classes.48 In addition, some structures are described by ordered sets, or vectors. There are different relationships between the types of sets – for instance, there is an injective function between prime sets and interval vectors. There are transformations that can be

---

46 See Van Der Walt, 2007.

47 Morris, Robert. (1989). Composition with Pitch Classes: A Theory of Compositional Design. *Music Theory Spectrum*, Vol 11-2, pp. 240-251.

48 Marvin, Elizabeth (2017). *A generalized theory of musical contour : its application to melodic and rhythmic analysis of non-tonal music and its perceptual and pedagogical implications* (Doctoral Dissertation). Retrieved from http://hdl.handle.net/1802/32238.

performed on sets or vectors, such as transposition or inversion, and some sets are symmetric relative to those transformations while some are not. All transformations done on these sets are typically assuming a mod-12 space. The problem with musical set theory, like combinatorics, is that it seems to work as a form of analysis only when it is also used in the process of generation; it is easy to find transposed pitch-class sets in atonal music which was developed with knowledge of set theory in mind, but the set theory analysis championed by the likes of Allan Forte doesn't seem to do much with tonal music. I believe that ultimately, what is called musical set theory is a specific theory that can be used in the analysis and generation of pieces of a certain style (as I used it in generating certain pieces), but the use of the type of set theory used for linguistic categorial grammars is much more compelling as a general meta-theory than traditional musical set theory.

Neural Networks

At this particular time in the history of computer science, it is inadvisable to attempt a new system for doing anything without either using neural networks or having a sophisticated reason not to use neural networks. At least for now, I fall into the second category. I will therefore explain why I do not believe neural networks can generate true music at this moment in time, and how I imagine neural networks could become useful in the future.

Neural networks have proved useful for many things, including tasks that we think of as uniquely human, such as speech and vision recognition, and strategical tasks, such as playing Go. However, neural networks have not yet cracked the problem of music. Current models seem to be about as capable as the Markov models used in this project – capable of generating music that sounds good at

the local level, but not globally coherent, and nowhere near as interesting musically as the pieces they were trained on.

Why might neural networks have trouble creating new music? There are several reasons. First of all, researchers are just starting to create neural network models which explicitly allow for storing, retrieving, and manipulating data structures.49 This may be a problem because good music requires manipulation of graph structures. Neural networks also don't do well with generalizing "program" knowledge; a program that is trained to perform transposition on 8 notes may fail to do so when presented with 16 notes (although again, this is changing as research evolves).50 Furthermore, all of the structures seen in music are relative. A "C4" note has absolutely no meaning by itself; its meaning is a complex function of the notes around it. This stands in contrast with linguistic neural models like *word2vec*, where, even if a word's inferred meaning (i.e., vector representation) is a function of it's surroundings, each individual word is assumed to have some individual meaning that the vector represents.

One way that I do think neural networks could be of use is in mining musical state-space – assuming that one is willing to analyze the given music in some way *before* giving it to a neural network. For instance, imagine training a piece of music on 1 billion short snippets of music as well as their precomputed *music2vec* representations. One could then possibly give the neural network a desired *music2vec* representation, and it would spit out a piece of music that is similar to that representation, even if the exact representation is not produceable.

Of course, creating little bits of music which satisfy given properties is only one part of the

---

49 Graves et al. (2014). Neural Turing Machines. Retrieved from https://arxiv.org/abs/1410.5401.

50 Graves et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature,* Vol 538, pp. 471–476.

problem of generating musical scores. One also needs to think about how all of these little pieces are going to connect (the syntax of the piece). There are two ways to go about this process. One could simply say that the overall structure is not a job for the neural network, while the filling in of the structure is. Alternatively, one could define a reward function for the kind of structures one might want to see. Although it is beyond the scope of this paper to discuss what such a reward function might be, it might involve the Kolmogorov complexity or entropy of the state-space. Then the total reward function for the neural network would be some function of how good its syntactic approach to state-space was, and how close the neural network could come to actually achieving it's desired route in state-space. However, whether this would actually be better than non-neural network approaches remains to be seen.

Constrained Models of Music

It is arguably harder to be a Western composer today than it has been in the history of music on this planet. Modern composers are supposed to be able to synthesize many different theories, and come up with some combination of these theories in a previously undiscovered form where the result is somehow "new." In contrast, in other cultures and at different periods of time music was much more constrained in how it sounds. Constraints make life easier for the composer, and they certainly make life easier for a programmer interested in generating music.

Mozart wrote an astounding 41 symphonies, compared to Beethoven's 9. While his were shorter than Beethoven's, most of Mozart's advantage lied in the strict conventions of the Classical era. The form of a sonata (the most common musical form) was strictly determined: A first theme would be presented in the home key. A second theme would then be presented in the dominant key. A transition

would lead back to the home key, and then the first and second themes would both be presented in the home key. Chord progressions were equally controlled, with only a few options available. There were a couple of styles of accompaniment to go with the melody, which was made up of short, repetitive motions.

It was fairly easy to generate a relatively good-sounding imitation of a Mozartean piece, which can be heard here: https://soundcloud.com/user-332259139/constrained-sonata. However, while this piece might be better-sounding to some than the other pieces on my website, it's a bit of a false pretense to call it generated music, as so much of it was pre-constrained – the accompaniment style, the chord progressions, the instrumental style, the form, etc. It's important to note that I was constraining the piece even more than Classical conventions, and that in Classical music, much of the interest lies in exploiting the few areas which aren't controlled by convention. However, my experiment convinced me that conventions designed to limit one's field of possibilities are powerful tools for speeding up the composing process.

Environments for Enhanced Musical Meaning

In addition to the meanings attributed to music above, there are certain usages and types of music which have additional meaning, such as in film music, vocal music, and musical sonification.

Film Music

Film music, one of the largest commercial sources of music, has been studied for the types of semantic content it needs to represent beyond what most music represents. Film music can be divided between diegetic music (which is assumed to be heard by the characters present as well as the

audience) and non-diegetic music (which is assumed to be heard by the audience alone). Both types of music serve specific purposes in the movie, although more research has been done on the purposes of non-diegetic music. According to Chattah, the music typically works semiotically by exploiting the cognitive process of metaphor-making; a particular aspect of the music is supposed to represent a particular aspect of the film.[51]

Virtually every musical feature can be and has been used in film music to symbolize something. Pitch is often raised to imply an increase in tension, or it may be used to represent physical height.[52] (One can argue whether "high-pitch=tension" is really a metaphor or a fact of biology, but this is somewhat irrelevant). Fast-paced music is generally played during car chases or other scenarios where physical speed is implied.[53] In some movies, timbre (the basic sound of an instrument, keeping pitch constant) is used to represent (or create?) tension.[54] In addition to these basic musical features, repertoire-specific musical objects can be used, such as employing ostinatos (incessantly repeating short figures, typical in certain Eastern cultures as well as in late 19th-early 20th century Western music) to represent tormenting thoughts or using deceptive cadences (a certain chord structure used at the end of the phrase that prevents the phrase from sounding finalized, typical in Classical era music) when there is a lack of closure.

Working with film music seems to problematize the approach of composing interacting musical objects. In the procedure I described above, it is possible not to know certain musical features. In fact, my experiments may have involved knowing more musical features than if I had taken the method to

---

[51] Chattah, Juan. (2006). *Semiotics, Pragmatics, and Metaphor in Film Music Analysis* (Doctoral Dissertation). Retrieved from http://diginole.lib.fsu.edu/islandora/object/fsu:182108

[52] See Chattah on "Tom and Jerry"

[53] See Chattah on "Crouching Tiger Hidden Dragons"

[54] See Chattah on "The Conversation"

its logical conclusion, and required combinators, higher order functions, and function currying to do most of the work (for example, treating contours as being generated by functions rather than as primitives in themselves). However, a film director wants to know all the salient features at a given point in time, as the basics (pitch height, major/minor, pace, tonal/atonal, etc.) will all be taken as symbols for plot elements. Thus, to automatically generate film music, one would need to constrain the musical program being generated to lie in a particular portion of feature-space.

Vocal Music

Vocal music poses particular challenges for the composer. Some of these challenges relate to the unique qualities of the voice. The voice can produce more timbres than most instruments, and so the composer must choose how the sound should be pronounced. In addition, the range of a voice is constrained in such a way that some vowels can be sung at a high range, while others can't. However, more relevant to this paper are challenges involving the syntax and semantics of words.

In regards to understanding what each word is supposed to be, challenges vary depending on what style of music and which language is being used.[55] Certain languages are tonal, meaning that in order to understand the words they have to be spoken in a certain pitch contour. Other languages are not tonal, but require placing the stress on a particular syllable. Some words are meant to be spoken quickly, such as “pope,” while others take more time, such as “home.”

There are challenges in conveying the structure and meaning of the words in a piece of vocal music beyond making sure that the words are heard. While tone-painting (using musical imagery that mimics the meaning of a word) is not as in vogue today as it was in the Renaissance, you would almost

---

[55] The source for this paragraph was an informal interview with a modern vocal composer, David Wolfson.

never expect to hear the word “high” being pronounced in a low pitch, unless the intended effect was irony. In addition, a fascinating study showed the benefit of using harmonically related material to depict semantically related words, which results in better recognition of the semantic relatedness.[56] Sometimes, stress is needed to interpret the correct overall meaning of the sentence. (Consider the difference in meaning of three possible answers to the question “How was class?” - “THIS class was fun”, “this CLASS was fun”, and “this class WAS fun.”) In other instances, pauses are needed to infer the correct syntactic structure without the help of punctuation; Nooteboom cites the example “the queen said the knight is a monster”, which can be interpreted as “the queen, said the knight, is a monster” or “the queen said, the knight is a monster.”[57]

As so little has been definitively concluded regarding the syntax of music, it is understandable that still less has been said about how it relates to the syntax of language when the two are paired. However, it is certain that the requirements of vocal music pose an even greater challenge to my method of “composing musical objects” than film music. Vocal musical restrictions not only require the music to have specific features overall, but require individual notes within the surface of the music to have a particular value. Thus, a program synthesis model of vocal music would have a much narrower range of acceptable outputs than the current model.

---

[56] Charronat et al. (2005). Musical structure modulates semantic priming in vocal music. *Cognition,* Vol 94(3).

[57] Nootebum, Sieb. (1997). The Prosody of Speech: Melody and Rhythm. *The Handbook of Phonetic Sciences*, Vol 5.

Data Sonification

Data sonification is the practice of somehow using non-musical data as parameters for a sound composition. Data sources that have been sonified include weather data,[58] data from particle accelerators,[59] cosmological data,[60] and financial data.[61] Some data sources have a natural transformation into sound (for instance, a sonification of seismic waves as sound waves), while others have no obvious mapping from data to sound. Data sonification is an umbrella term that includes compositions designed to present information much like graphs present information, as well as compositions which are intended to produce a more artistic effect. In practice, there is often a tradeoff between the degree to which a sonification clearly presents the nature of the data and the degree to which the final product is perceived as musical.

In one experiment with sonification, I attempted to sonify the algorithm "mergesort." Starting with a randomly generated list of notes, I called the recursive function, writing the partially sorted list of notes to the score during every merge. Adding differences in duration for merging from list 1 and list 2 highlighted the use of this particular sorting strategy, and the end result of a sorted list of pitches by height is clearly audible. The result can be heard here: https://soundcloud.com/user-332259139/mergesort. In this case, the semantics of the piece is entirely defined by one function - namely, mergesort – with perhaps a few additional parameters to turn a non-musical algorithm into a musical

---

58 Lindroth, Scott. *Sonification of Global Temperatures, 1850 – Aug 2012.* Retrieved from http://people.duke.edu/~scott1/climate.html

59 Jarlett, Harriet. (2016). *Sonified Higgs Data Shows a Surprising Result.* Retrieved from https://home.cern/about/updates/2016/04/sonified-higgs-data-show-surprising-result

60 McGee, Ryan et al. *Sonifying the Cosmic Microwave Background.* Retrieved from https://www.mat.ucsb.edu/res_proj7.php

61 Janata, Petr and Childs, Edward. Marketbuzz: Sonification of Real-Time Financial Data. Retrieved from https://atonal.ucdavis.edu/publications/papers/janata_and_childs_icad_2004.pdf

one. One can still use the “musical object” paradigm to describe this piece – it's simply an extremely low complexity piece of music which is derived from one musical object.

The second piece I wrote is more difficult to analyze semantically. I took Martin Luther King Jr's “I Have a Dream” speech, and ran a grammatical analysis of two sentences (the result was a tree structure with the sentence as the root node and parts of speech as the leaves). I then assigned melodies to each part of speech, accompaniment instruments to each part of speech, and durational patterns to every tree depth. Spanning from the first word of the first sentence to the last word of the second sentence, every token was translated into a measure of music. The result (which can be heard here: https://soundcloud.com/user-332259139/i-have-a-dream-sonification), while less coherent than the average piece, does not obviously sound like a sonification, and in fact, a lot of parameters were left to choice or chance. However, to some extent there is data that can be inferred from the composition. For instance, one can hear when the depth of the parse tree is relatively deep, and one can get a sense of the distribution of parts of speech. Ultimately, the question of whether the syntax of King's speech is part of the syntax or semantics of the resulting piece is open for interpretation.

Film music, vocal music, and data sonification require additional explanation from a computational perspective because they require additional restrictions to generate, and present an interesting problem with regards to discussing musical meaning. In order to generate film and vocal music automatically, or to generate both informative and musical sonifications, it will be necessary to rethink the musical object framework to consider additional features beyond the ones strictly needed to create the music. Models of musical semantics in film and sonification require inquiry beyond the

scope of this paper: Can research in narrative grammar be coupled with research in musical grammar?[62] How faithful to data does music have to be before it is a part of the musical semantics? I look forward to seeing more research in these areas.

---

[62] Greimas, A. and Porter, C. Elements of a Narrative Grammar. *Diacritics,* Vol 7-1, pp. 23-40.

Conclusion

Over 2500 years ago, Pythagoras declared that music is "sounding number," or math itself. At the same time, at least since the Renaissance, music theorists have viewed even instrumental music as being somehow akin to language, with meaning and grammatical structure. Today, we have determined that math and formal methods are not orthogonal to the study of language, but rather enhance it. Slowly, researchers are realizing that the methods we have developed to talk about language can also be used to bridge the gap between mathematical and linguistic views of music, and to get at the heart of what musical meaning is. I believe there are several promising formalisms which provide ways to explore the syntax and semantics of music from Western as well as non-Western cultures. These formalisms will include probabilistic methods, generative grammars in the style of Chomsky, the syntax and semantics of musical state space, musical type theory and lambda calculus, and more. The details of how these mathematical models produce meaning may, depending on the model, vary based on culture or semiotic purpose of the music; however, in every case it seems clear that music does have a syntax and semantics, however hard to define.

References

Bartel, Dietrich. (1997) *Musica Poetica.* University of Nebraska Press.

Solomon, Maynard. (2004) *Late Beethoven: Music, Thought, Imagination.* University of California Press.

Dunton, Green. (1930). Schopenhauer and Music. *The Music Quarterly,* Vol 16, pp 199-206.

Kimmey, Roy. (2009). The Critic in Question: Eduard Hanslick and Viennese Musical Identity. *The Harvard College History Review*, Vol X, Issue 1.

Simms, Bryan. (2000) *The Atonal Music of Arnold Schoenberg 1908-1923.* Oxford University Press.

Morris, Robert. (2004-2005). Aspects of Post-Tonal Canon Systems. *Integral*, Vol 18-19, pp 189-221.

Van der Walt, Salome. (2007). *Rhythmic Techniques in A Selection of Oliver Merriaen's Piano Works* (master's thesis). Retrieved from http://www.repository.up.ac.za/

Zhang, Ada. (2009). *The Framework of Music Theory as Represented With Groups.* Retrieved from https://sites.math.washington.edu/~morrow/336_09/papers/Ada.pdf.

Jiang, Tao; Li, Ming; Ravikumar, Bala; and Regan, Kenneth. *Formal Grammars and Languages.* Retrieved from http://www.cs.ucr.edu/~jiang/cs215/tao-new.pdf

Bernstein, Leonard. (1973). *The Unanswered Question: Six Talks at Harvard*. Cambridge, Massachusetts: Harvard University Press.

Lerdahl, Fred and Jackendoff, Ray. (1981). *A Generative Theory of Tonal Music.* The MIT Press.

Harmonic Syntax – The Idealized Phrase, retrieved from http://openmusictheory.com/harmonicSyntax1.html

Kunert R, Willems RM, Casasanto D, Patel AD, Hagoort P (2015). Music and Language Syntax Interact in Broca's Area: An fMRI Study. *PLoS ONE* 10(11): e0141069. doi:10.1371/journal.pone.0141069

Koelsch. (2005). Neural substrates of processing syntax and semantics in music. *Curr Opin Neurobiol.* Vol 207-12

Wiggins, Geraint. (1998). Syntax, music, and the meaning of meaning. *Proceedings of the First Symposium on Music and Computers*. pp. 18–23.

Brown S, and Jordania J. (2011) Universals in the world's musics. *Psychology of Music.* Vol 41(2), pp. 229-248.

Temperly, David. (2006). *Music and Probability.* The MIT Press.

Pearce MT and Wiggins GA. (2007). Evaluating cognitive models of musical composition. *Proceedings of the 4th International Joint Workshop on Computational Creativity*. Pp 73-80.

Draper, Steve. (2015). *Creativity.* University of Glasgow. Retrieved from http://www.psy.gla.ac.uk/~steve/best/creative.html.

Quick, Donya and Hudak, Paul. (2013). Grammar-Based Automated Music Composition in Haskell. *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design,* pp 59-70.

Wilding, MT. (2013). *Harmonic Analysis of Music Using Combinatory Categorial Grammar* (doctoral thesis). Retrieved from https://www.era.lib.ed.ac.uk/handle/1842/8019

Manousakis, Stelios. (2006). *Musical L-Systems* (Master's Thesis). Retrieved from http://carlosreynoso.com.ar/archivos/manousakis.pdf

Biles, John. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. *Proceedings of the 1994 International Computer Music Conference*.

Matic, Dragan. (2010). A Genetic Algorithm for Composing Music. *Yugoslav Journal of Operations Research,* Vol 20, pp. 157-177.

Jeong, Jae. (2015). Automatic Evolutionary Music Composition Based on Multi-objective Genetic Algorithm. *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems.* Vol 2.

Mikolov, Tomas et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in neural information processing systems***.**

Kiros, Ryan et al. (2015). *Skip-Thought Vectors.* Retrieved from https://arxiv.org/abs/1506.06726.

Cuthbert, Scott, and Ariza, Cristopher. (2010). music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. *International Society for Music Information Retrieval,* pp 637-642.

Taube, Heinrich. (2004). *Notes from the Metalevel: Introduction to Algorithmic Composition.* Netherlands, Swets & Zeitlinger Lisse.

Sumit Gulwani. (2010). Dimensions in Program Synthesis. *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming,* pp 13-14.

Rajeev Alur et al. (2013). Syntax Guided Synthesis. *FMCAD,* pp. 1-17.

Christoph Kreitz. (1998). Program Synthesis. *Automated Deduction – A Basis for Applications*, pp. 105–134 .

C Mohring, B Werner. (1993). Synthesis of ML Programs in the System Coq. *Journal of Symbolic Computation* Vol 15-5-6.

Lim, J., & Yoo, S. (2016). Field Report: Applying Monte Carlo Tree Search for Program Synthesis. *Search-Based Software Engineering*. Springer International Publishing.

Osera and Zdancewic. (2016). Type- and Example-Driven Program Synthesis. *Symposium on Trends in Functional Programming.*

Chandler, Paul. (2005). *Music and Schema Theory.* Retrieved from http://tzone.org/~okko/html/documents/music_and_cognition.pdf

Read, R. (1997). Combinatorial Problems in the theory of music. *Discrete Math*, Vol 167-168, pp. 543-551.

Nolan, Catherine. (2000). On Musical Space and Combinatorics: Historical and Conceptual Perspectives in Music Theory. *Bridges – Mathematical Connections in Art, Music, and Science.*

Morris, Robert. (1989). Composition with Pitch Classes: A Theory of Compositional Design. *Music Theory Spectrum*, Vol 11-2, pp. 240-251.

Marvin, Elizabeth (2017). *A generalized theory of musical contour : its application to melodic and rhythmic analysis of non-tonal music and its perceptual and pedagogical implications* (Doctoral Dissertation). Retrieved from http://hdl.handle.net/1802/32238.

Graves et al. (2014). Neural Turing Machines. Retrieved from https://arxiv.org/abs/1410.5401.

Graves et al. (2016). Hybrid computing using a neural network with dynamic external memory.

Nature, Vol 538, pp. 471–476.

Chattah, Juan. (2006). *Semiotics, Pragmatics, and Metaphor in Film Music Analysis* (Doctoral Dissertation). Retrieved from http://diginole.lib.fsu.edu/islandora/object/fsu:182108

Charronat et al. (2005). Musical structure modulates semantic priming in vocal music. *Cognition,* Vol 94(3).

Nootebum, Sieb. (1997). The Prosody of Speech: Melody and Rhythm.
The Handbook of Phonetic Sciences, Vol 5.

Lindroth, Scott. *Sonification of Global Temperatures, 1850 – Aug 2012.* Retrieved from http://people.duke.edu/~scott1/climate.html

Jarlett, Harriet. (2016). *Sonified Higgs Data Shows a Surprising Result.* Retrieved from https://home.cern/about/updates/2016/04/sonified-higgs-data-show-surprising-result

McGee, Ryan et al. *Sonifying the Cosmic Microwave Background.* Retrieved from https://www.mat.ucsb.edu/res_proj7.php

Janata, Petr and Childs, Edward. Marketbuzz: Sonification of Real-Time Financial Data. Retrieved from https://atonal.ucdavis.edu/publications/papers/janata_and_childs_icad_2004.pdf

Greimas, A. and Porter, C. Elements of a Narrative Grammar. *Diacritics,* Vol 7-1, pp. 23-40.

Listing of Audio Examples

Markov Model: https://soundcloud.com/user-332259139/markov-model-music

Context-Free Harmonic Grammar: https://soundcloud.com/user-332259139/context-free-grammar-music

Lindenmayer System: https://soundcloud.com/user-332259139/lindemayer-system-music

Musical Object Composition: https://soundcloud.com/user-332259139/messiaen-like-generated-music

Program Synthesis Composition 1: https://soundcloud.com/user-332259139/program-synthesis-music

Program Synthesis Composition 2: https://soundcloud.com/user-332259139/program-synthesis-music-2

Constrained Sonata Composition: https://soundcloud.com/user-332259139/constrained-sonata

Mergesort Sonification: https://soundcloud.com/user-332259139/mergesort

“I have a Dream Sonification”: https://soundcloud.com/user-332259139/i-have-a-dream-sonification