MUME 2016 - The Fourth International Workshop on Musical Metacreation, ISBN #978-0-86491-397-5

Towards the Generation of Melodic Structure

Ryan Groves
groves.ryan@gmail.com

Abstract

This research explores the generation of melodies through the creation of a probabilistic analytical model of melodies. Using a Natural Language Processing technique utilized for the automatic reduction of melodies: the Probabilistic Context-Free Grammar (PCFG), it is possible to reduce new incipit melodies, or to generate and embellish melodies. Automatic melodic reduction has been previously explored by means of a probabilistic grammar (Gilbert and Conklin 2007) (Abdallah and Gold 2014). However, these methods used unsupervised learning to estimate the probabilities for the grammar rules rather than a corpus-based evaluation. A treebank of analyses using the Generative Theory of Tonal Music (GTTM) exists (Hamanaka, Hirata, and Tojo 2007), which contains 300 Western tonal melodies and their corresponding melodic reductions in tree format. In this work, a new representation of the GTTM grammar is created using a higher-level representation based on intervals. Then, supervised learning is used to train a PCFG on the treebanks with different versions of the new data representation. The resulting model is evaluated on its ability to create accurate reduction trees, and its ability to generate melodies is subjectively explored. With this approach, each generated melody will not only be a sequence of pitches, but also a hierarchical structure containing the melodic reduction of the generated melody. Multiple data representations are tested, and example output reductions—and embellishments—are shown. Based on the comparison of output trees with their corresponding solution trees, the best-performing PCFG was the one trained on prolongational reductions.

Introduction

Melodic reduction is the process of finding the more structural notes in a melody. Through this process, notes that are deemed less structurally important are systematically removed from the melody. This work aims to model the process of reducing melodies using probabilistic methods, and to then reverse the process to generate or embellish new melodies. First, it is important to understand the music theoretical background of utilizing tree structures for melodic reduction.

Composers have long used the rules of ornamentation to elaborate certain notes. In the early 20th century, music theorist Heinrich Schenker developed a hierarchical theory of music reduction (a comprehensive list of Schenker's publications was assembled by David Beach (Beach 1969)). Schenker ascribed each note in the musical surface as an elaboration of a representative musical object found in the deeper levels of reduction.

In the 1980s, another theory of musical reduction was detailed in GTTM (Lerdahl and Jackendoff 1983). The authors' goal was to create a formally-defined generative grammar for reducing a musical piece. In GTTM, every musical object in a piece is *subsumed* by another musical object, which means that the subsumed musical object is directly subordinate to the other. In detailing this process, Lerdahl and Jackendoff begin by breaking down metrical hierarchy, then move on to identifying a grouping hierarchy (separate from the metrical hierarchy). Finally, they create two forms of musical reductions using the information from the metrical and grouping hierarchies—the time-span reduction, and the prolongational reduction. The former details the large-scale grouping of a piece, while the latter notates the ebb and flow of musical tension in a piece.

Many researchers have taken the idea—inspired by GTTM or otherwise—of utilizing formal grammars as a technique for reducing or even generating music (detailed in Section). A dataset for the music-theoretical analysis of melodies using GTTM has been created in the pursuit of implementing GTTM as a software system (Hamanaka, Hirata, and Tojo 2007). This dataset contains 300 Western classical melodies with their corresponding reductions, as notated by music theorists educated in the principles of GTTM. Each analysis is stored as a tree structure, which id directly compatible with computational grammars and their corresponding parse trees. The GTTM dataset is the corpus used for the supervised PCFG detailed in this paper.

This work was inspired by previous research on a PCFG for melodic reduction (Gilbert and Conklin 2007), in which a grammar was designed by hand to reflect the common melodic movements found in Western classical music, based on the compositional rules of ornamentation. Us-

The Fourth International Workshop on Musical Metacreation, MUME 2016.
www.musicalmetacreation.org.

ing that hand-made grammar, the researchers used a dataset of melodies to calculate the probabilities of the PCFG using unsupervised learning. This research aims to simulate and perform the process of melodic reduction, using a supervised Probabilisitic Context-Free Grammar (PCFG). Furthermore, this work explores the possibilities of generating new melodies with the probabilistic model of hierarchical melodic structure. By utilizing a ground-truth dataset, it is possible to directly induce a grammar from the solution trees, creating a unique set of production rules and modelling the probabilities for each rule expansion. Different data representations will be explored and evaluated based on the accuracy of their resulting parse trees. A standard metric for tree comparison is used, and example melodic reductions will be displayed.

The structure of this paper is as follows: The following section provides a brief history of implementations of GTTM, as well as an overview of formal grammars used for musical purposes. Section 3 presents the theoretical foundations of inducing a probabilistic grammar. Section 4 describes the data set that will be used, giving a more detailed description of the data structure available, and the different types of melodic reductions that were notated. Section 5 describes the framework built for converting the input data type to an equivalent type that is compatible with a PCFG. Section 6 presents the experiment, giving a detailed account of the data representations used, as well as the comparison and evaluation method, and the results of the different tests performed. Section 6 also provides some experimental work in modelling harmony. Section 7 provides some closing remarks.

Literature Review

In order to reduce a melody, a hierarchy of musical events must be established in which more important events are at a higher level in the hierarchy. Methods that create such a structure can be considered to be in the same space as melodic reduction, although some of these methods may apply to polyphonic music as well. The current section details research regarding hierarchical models for symbolic musical analysis.

Grammars in Music

In 1979, utilizing grammars for music was already of much interest, such that a survey of the different approaches was in order (Roads and Wieneke 1979). Ruwet (Ruwet 1975) suggested that a generative grammar would be an excellent model for the creation of a top-down theory of music. Smoliar (Smoliar 1976) attempted to decompose musical structure (including melodies) from audio signals with a grammar-based system.

Baroni et al. (Baroni et al. 1982) also created a grammatical system for analyzing and generating melodies in the style of Lutheran chorales and French chansons. The computer program would create a completed, embellished melody from an input that consisted of a so-called "primitive phrase" (Baroni et al. 1982, 208).

Baroni and Jacoboni designed a grammar that was utilized to analyze and generate melodies in the style of major-mode chorales by Bach (Baroni and Jacobini 1975; Baroni and Jacoboni 1978). The output of the system would generate the soprano part of the first two phrases of the chorale.

Probabilistic Grammars

Gilbert and Conklin (Gilbert and Conklin 2007) designed a PCFG for melodic reduction and utilized unsupervised learning on 185 of Bach's chorales from the Essen Folksong Collection. This grammar was also explored by Abdallah and Gold (Abdallah and Gold 2014), who implemented a system in the logical probabilistic framework PRISM for the comparison of probabilistic systems applied to automatic melodic analysis. The authors implemented the melodic reduction grammar provided by Gilbert and Conklin using two separate parameterizations and compared the results against four different variations of Markov models. The evaluation method was based on data compression. Tested over four separate subsets of the Essen Folksong Collection, the authors found that the grammar designed by Gilbert and Conklin was the best performer with 2.68 bits per note over all the datasets. The same authors also collaborated with Marsden (Abdallah, Gold, and Marsden 2016) to detail an overview of probabilistic systems used for the analysis of symbolic music, including melodies.

Hamanaka et al. also used a PCFG for melodic reduction (Hamanaka, Hirata, and Tojo 2015). The authors used the dataset of treebanks that they had previously created (Hamanaka, Hirata, and Tojo 2007) to run supervised learning on a custom-made grammar that he designed, in order to automatically generate time-span reduction trees. This work is very similar to the work presented here, with two exceptions. First, the grammar was not learned from the data. Secondly, Hamanaka used a series of processes on the test melodies using previous systems he had built. This is in contrast with the current work, which attempted to encapsulate the GTTM process solely in a grammar. These systems notated the metrical and grouping structure of the input melody, before inputting that data into the PCFG. Hamanaka achieves a performance of 76% tree accuracy.

Similar Methods for Musical Reduction

Creating a system that can perform a musical reduction according to the theory of Heinrich Schenker has also been the topic of much research. Marsden explored the use of Schenkerian reductions for identifying variations of melodies (Marsden 2010). PCFGs have not yet been utilized for this particular task. One notable caveat is the probabilistic modelling of Schenkerian reductions, using a tree-based structure (Kirlin 2014). Kirlin did not explicitly use a PCFG, however his model was quite similar, and also was a supervised learning method.

Supervised Learning of a PCFG

Grammars were formalized by Chomsky (Chomsky 1956) and extended by himself (Chomsky 1959) and Backus et al. (Backus 1959). Grammars are composed of a series of production rules, which specify the relationships between

terminals (tokens like 'dog' or 'walk') and non-terminals (which can be expanded in to sequences of non-terminals or terminals). Each production rule has a right-hand side, that represents the expansion of the term found on the left-hand side. In a Context-Free Grammar (CFG), the left-hand side consists of a single non-terminal, and the right-hand side consists of a sequence of non-terminals and terminals. Given a CFG and an input sequence of terminals, the sequence can be *parsed*, creating a hierarchical structure by recursively finding all applicable rules, until a tree structure is formed.

Probabilistic CFGs extend the CFG by also modelling the probabilities of each right-hand side expansion for every production rule. The sum of probabilities for all of the right-hand side expansions of each rule must sum to 1, although in practice this constraint is sometimes loosened.

Once a PCFG is calculated, it is possible to find the *most probable* parse tree, by cumulatively multiplying each production rule's probability throughout the tree, for every possible parse tree.

Inducing a PCFG

When a set of parse tree solutions (called a *treebank*) exists for a particular set of input sequences, it is possible to construct the grammar directly from the data. In this process, each parse tree from the treebank will be broken apart, so that the production rule at every branch is isolated. A grammar will be formed by accumulating every rule that is found at each branch in each tree, throughout the entire treebank. This new grammar is not identical to the grammar of GTTM, but an equivalent representation that is more conducive to melodic generation. When a rule and its corresponding expansions occurs multiple times, the probabilities of the right-hand side expansion possibilities are modelled. Inducing a PCFG is a form of supervised learning.

GTTM Dataset

The GTTM dataset contains the hierarchical reductions (trees) of melodies in an Extensible Markup Language (XML) representation.

There are two different types of reduction trees that are created with the theories in GTTM: time-span reduction trees, and prolongational reduction trees. The time-span reduction is built upon the grouping structure analysis provided in GTTM, which in turn uses the metrical structure analysis to influence its decision-making. Time-span reduction trees are generally more reliant on the metrical information of a piece, since they utilize the grouping structure directly. The prolongational reductions are designed to notate the ebb and flow of tension and progression in a piece. In fact, in GTTM, the prolongational reductions use time-span reduction trees as a starting point, but then build the branching system from the top, down, based on pitch and harmonic content in addition to the time-span information.

The entire dataset consists of 300 melodies, with analyses for each. However, the prolongational reduction trees are only provided for 100 of the 300 melodies, while the time-span trees are provided for all 300 melodies. The prolongational reductions require the annotations of the underlying harmony. Likewise, there are only 100 harmonic analyses in the dataset.

Forming the PCFG

When learning a grammar directly from a dataset of annotations, the most important decision to make is the data representation. The representation chosen should be able to capture the most relevant characteristics of the data. Similar to Gilbert and Conklin (Gilbert and Conklin 2007), each rule models two consecutive intervals in a sequence of three notes, and had the following form (notes labelled as *n1* through *n3*):

$$interval_{n1,n3} \rightarrow interval_{n1,n2} \;\; interval_{n2,n3} \quad (1)$$

The motivation was that melodic rules often involve a sequence of 3 notes. This is true for the passing tone, neighbor tone, cambiata, and the escape tone. The repetition rule would normally require only two notes, however to keep a consistent format, repetitions were only reduced when three consecutive notes of the same pitch were found, which were then reduced to two notes of the same pitch (creating one interval). This form of one interval expanding into two consecutive intervals for the grammatical rules was adopted for this research.

A Framework for Converting Trees

Utilizing a representation that required a sequence of two intervals in every right-hand expansion presented a problem, because the GTTM reduction trees were in a format that associated pairs of notes at each branch intersection—not the three consecutive notes required for the two consecutive intervals. Given this challenge, a framework was developed to convert the note representation of the GTTM data into the interval notation desired, and to build the corresponding tree structure using the interval representation.

An example GTTM tree is shown in Figure 1. Note that at the end of every branch is a single note. An algorithm was developed to allow the conversion of these note-based trees to any interval representation desired, based on a sequence of 3 notes. The algorithm traverses the tree from the top, down, in a breadth-wise fashion. At each level of depth, the sequence of notes at that depth are broken into sets of 3 consecutive notes, and their intervals are computed. Figure 2 highlights the breadth-wise traversal process.

The framework was built in Python[1]. It takes a function as input, which allows the user to define unique interval representations.

Training/Induction

The Python-based Natural Language Toolkit (NLTK) was used for the process of PCFG induction (Loper and Bird 2002). Given a treebank of solutions, the process for inducing a PCFG is described as follows. For every tree in the treebank, traverse through the tree to identify each branching location. For every branching location, create a rule with

[1] https://github.com/bigpianist/SupervisedPCFG_MelodicReduction

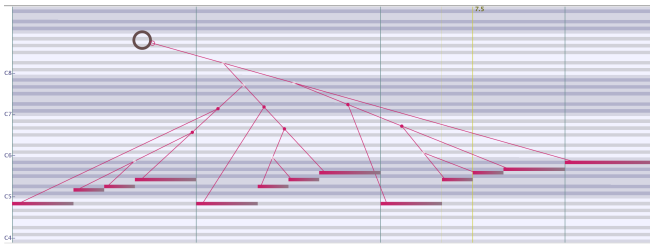


Figure 1: The prolongational reduction tree for half of the first melody in the GTTM dataset, Frédéric Chopin's "Grande Valse Brillante", as displayed in the GTTM visualizer provided by Hamanaka, Hirata, and Tojo (Hamanaka, Hirata, and Tojo 2007).

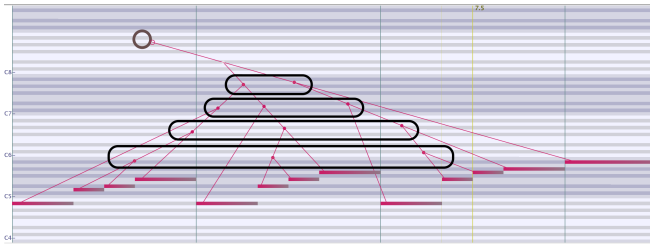


Figure 2: A depiction of the process for converting a tree that uses a note representation to a tree that uses an interval representation, by traversing the tree breadth-wise and relating sets of 3 notes.

the node label as the left-hand side, and the children as the right-hand side. Collect the set of rules found at every branch of every tree in the treebank, and pass that list of production rule instances into NLTK's induce_pcfg function. The induce_pcfg function will catalogue every rule, and build up a grammar based on those rules. It will also model the probability of each rule's unique expansions.

The Experiment

Given a method for creating a treebank with any interval-based data representation from the GTTM dataset and inducing the corresponding PCFG, an experiment was designed to test the efficacy of different data representations when applied to the process of melodic reduction. This section details the experiment that was performed. First, different representations that were tested are presented. Then, the comparison and evaluation method are described. After that, the results of cross-fold evaluation for the PCFG created with each different data representation are shown. Finally, some experimental work for modelling harmony with grammar is presented.

Data Representations

For the representation of intervals between two consecutive notes, this research focused on a few certain musical attributes. These attributes were tested first in isolation, and then in combination. The following descriptions relate to the attributes labelled in the results table (the key for each attribute is given in parentheses following the name).

Pitch The difference in pitch between two notes was a part of every data representation tested. However, the encodings for these pitch values varied. Initially, a simple pitch class representation was used. This allowed pitch intervals at different points in the musical scale to be grouped into the same production rules. It was assumed that direction of pitch would also be an important factor, so the **pitch class (PC)** attribute allowed the following range of intervals: [-11, 11]. Melodic embellishment rules often apply to the same movements of intervals within a musical scale. For this reason, the **Key-Relative pitch class (KPC)** was also used, which allowed a range of intervals from [-7, 7], measuring the distance in diatonic steps between two consecutive notes.

Metrical Onset For encoding the metrical relationships between two notes, the *metric delta* representation was borrowed from previous research (Gilbert and Conklin 2007). This metric delta assigns every onset to a level in a metrical hierarchy. The metrical hierarchy is composed of levels of descending importance, based on their onset location within a metrical grid. The onsets were assigned a level based on their closest onset location in the metrical hierarchy.

Because the GTTM dataset contains either 100 or 300 solutions (for prolongational reduction trees and time-span trees, respectively), the data representations had to be designed to limit the number of unique production rules created in the PCFG. With too many production rules, there is an increased chance of production rules that have a zero probability, which results in the failure to parse certain test melodies. Therefore, two separate metrical onset attributes were created. One which represented the full metrical hierarchy, named **Metric Delta Full (Met1)**, and one which represented only the directional change in metric level, named **Metric Delta Reduced (Met0)**.

Harmonic Relationship This research was also designed to test whether or not the information of a note's relationship to the underlying harmony was useful in the melodic reduction process. A **Chord Tone Change (CT)** attribute was therefore created, which labelled whether or not each note in the interval was a chord tone. This created four possibilities: a chord tone followed by a chord tone, a chord tone followed by a non-chord tone, a non-chord tone followed by a chord tone and a non-chord tone followed by a non-chord tone. This rule was designed to test whether harmonic relationships affected the reduction process.

Comparison

The comparison method chosen was identical to the methods used in other experiments of the same type, in which the output of the system is a tree structure, and the tree solutions are available (Hamanaka, Hirata, and Tojo 2007; **?**). First, for a given test, the input melody is parsed, which yields the most probable parse tree as an output. The output trees are then compared with the solution trees. To do so, the tree is simply traversed, and each node from the output

tree is compared for equivalence to the corresponding node in the solution tree. This method is somewhat strict, in that mistakes towards the bottom of the tree will be propagated upwards, so incorrect rule applications will be counted as incorrect in multiple places.

Evaluation

Cross-fold evaluation was used to perform the evaluation. The entire treebank of solutions were first partitioned into 5 subsets, and 1 subset was used for the test set in 5 iterations of the training and comparison process. The results were then averaged. In order to keep consistency across data representations, the same test and training sets were used for each cross-validation process.

Results

Each data representation that was selected was performed on both the set of time-span reduction trees and the set of prolongational reduction trees, when possible. As mentioned previously, the set of prolongational reduction trees amounted to only 100 samples, while the time-span trees amounted to 300. In some situations, the data representation would create too many unique production rules, and not all the test melodies could be parsed. All of the data representations in the results table had at least a 90% coverage of the test melodies, meaning that at least 90% of the tests could be parsed and compared. There are also two data representations that use time-span trees with the harmonic representation. For these tests, the solution set contained only 100 samples as opposed to the usual 300 for time-span trees, since there is only harmonic information for 100 of the 300 melodies.

Tree-type	PC	KPC	Met1	Met0	CT	% nodes correct
TS	X					35.33
PR	X					38.57
TS		X			X	40.40
PR		X			X	38.50
TS		X	X			44.12
PR		X		X		46.55
TS		X		X	X	44.80
PR		X		X	X	46.74

Table 1: The effectiveness of each of the PCFG models, with the details of their corresponding data representation

These results mostly progress as one might expect. Looking at only the tests done with time-span trees, the results improve initially when using the **Key-Relative pitch class** encoding for pitch intervals paired with the **Chord Tone Change** feature; it received a 5% increase as compared with the PCFG that only used the **pitch class** feature (which could be considered a baseline). It gained an even bigger increase when using the **Metric Delta Full** feature, an almost 9% increase in efficacy compared with the **pitch class** test. Combining metric and chord features with the **Key-Relative pitch class** encoding did not provide much further

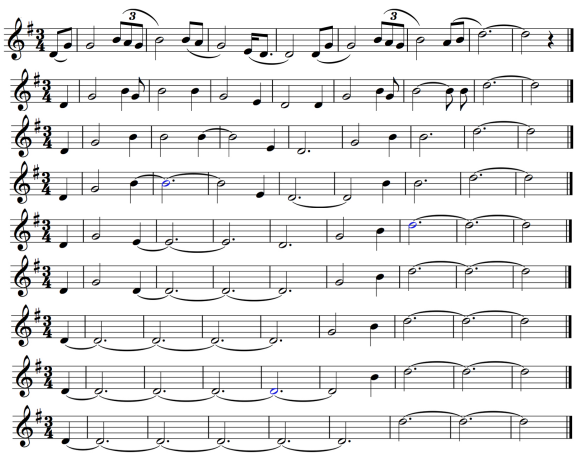


Figure 3: A set of melodies that show the progressive reductions, using the data representation that includes key-relative pitch class, metric delta and chord tone features.

gain that with the metric feature alone. The prolongational reduction also improved when given the metric delta information, however the harmonic relationship feature affected the outcome very little.

The best performing PCFG was induced from the prolongational reduction trees, and used a data representation that included the **Key-Relative pitch class** encoding combined with both the simplified metric delta and the chord tone information.

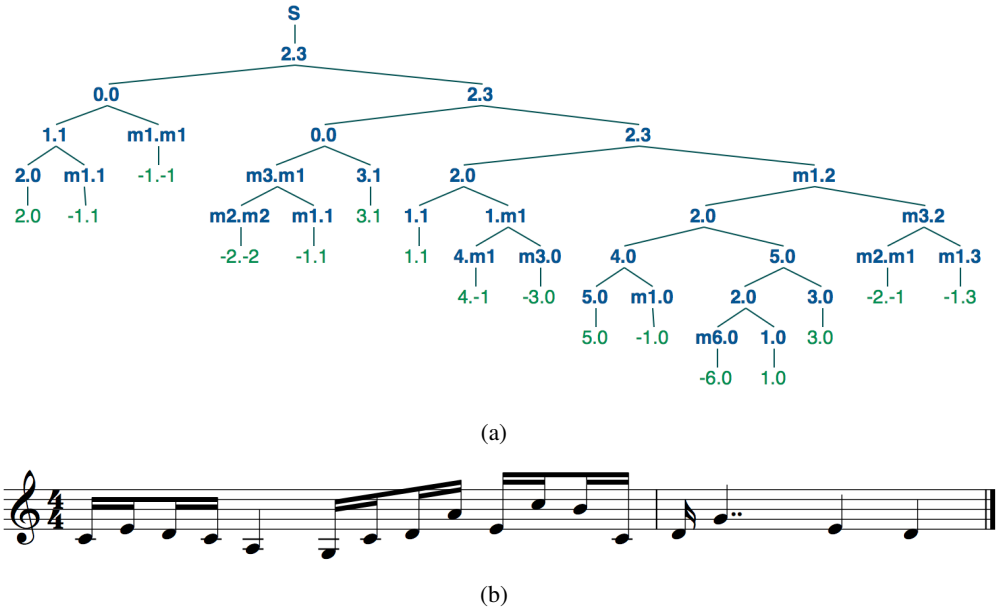
A specific reduction example helps to illustrate both the effectiveness and the drawbacks of using the induced PCFG for melodic reduction. Figure 3 displays the iterative reductions applied by pruning a PCFG tree, level by level. The grammar used to create this reduction was trained on prolongational reduction trees, and included the **Key-Relative pitch class** intervals, with notations for the **Metric Delta Reduced** feature, and the **Chord Tone Change** feature. This PCFG was the best performing, according to the evaluation metric. From a musicological perspective, the PCFG initially makes relatively sound decisions when reducing notes from the music surface. It is only when it begins to make decision at the deeper levels of reduction that it begins to choose the incorrect notes as the more important tones.

Melodic Generation

With a trained PCFG, one can sample from the probabilistic distributions in order to iteratively build a new tree structure representing a melody. With this process, each rule's set of right-hand side expansions are randomly sampled, starting with the Start rule. Depending on the data representation used, the resulting tree will contain the information for each transition between pitches, as well as each metric interval. Some example generated melodies are provided in Figure 5.

Generated Examples

The best-performing PCFG was the one trained on the prolongational reductions, and the data representation included



(a)

(b)

Figure 4: A generated tree with its corresponding melody.

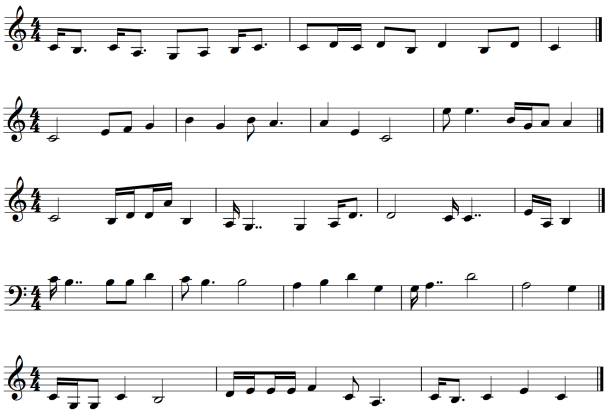


Figure 5: Some example generated melodies.

key-relative pitch classes, simple metric intervals, and chord tone information. In order to use the chord tone information, one would need to create a model for the harmonic sequences, and how they relate to the melodies–it would not be useful to simply assign a chord to each chord tone without some sense of harmonic progression as well as insight into which chord is appropriate for the given pitch. Similarly, using the *Metric Delta Reduced* representation for the onsets of the generated notes limits the rhythmic possibilities, because the onsets can only occur either at the same metric level, one metric level greater or one metric level smaller than the previous note. Using the **Metric Delta Full** data representation would therefore be advantageous because the onset possibilities are a lot more diverse. Because of these limitations, the melodies were generated from a PCFG that performed similarly well, but also had the advantage of using the **Metric Delta Full** encoding for onsets, the **Key-Relative Pitch Class** for pitches, and included no chord tone information. This representation had a 44.12% accuracy in forming melodic reduction trees (see Table 1).

Since the representation is formed with intervals, each leaf node of the tree is therefore relative to the previous note. Due to this representation choice, the first note of every generated melody must be assigned before the rest of the notes can be generated. For the sake of simplicity, each melody was generated in the key of C Major, with the tonic as the first note. Similarly, a quarter tone value was considered the first metric level, and each onset thereafter was generated based on the next possible onset in time that fell in that metric level.

An example melody is shown in Figure 4b. The tree structure that was used to generate this melody, which also represents the melody's reduction, is shown in Figure 4a. It is important to understand the format of the data representation. Each node is represented by the combination of the encoded features, each separated by a dot. A label of '1.2', for example, specifies that the key-relative pitch class is a positive interval of 1 from the previous node, and the metric interval is a positive interval of 2. The internal nodes have the same label format as the leaf nodes. Furthermore, each of the features in the child nodes should add up to the corresponding feature in the given parent node.

The first four notes of the melody are a good example of how the embellishment rules can manifest. At the top of this sub-tree is the node label '0.0'. This means that the pitch should repeat at the same onset interval. After embellishment, it can be seen that this label represents the interval between note 1 and note 4 in the melody, both at the metric level of a 1/16th note (this melody was generated with the

initial note starting on the tonic, at the 1/16th note metric level). The Repeat Rule is then embellished with two child nodes that together compose a Neighbor Tone Rule. The first rising step of the Neighbor Tone Rule is then further embellished by an Escape Tone Rule, completing the 4-note figure. This type of figure is common, and can be formed using alternative embellishment rules. For example, it could also be formed with a jump of 2 diatonic steps, followed by a descending Passing Tone Rule. What this particular formation in the example could indicate is that the Neighbor Tone (the third note in the figure) is more structurally important than the initial jump (the second note).

Future Work: Harmonic Grammar

The lack of the ability to generate notes that relate to an underlying harmony is a serious drawback of the model. The amount of ground truth data that was available in the GTTM dataset limited the complexity of the data representations such that it was only possible to encode a simple boolean feature that specified whether or not a note was part of the underlying harmony.

As a first step towards bridging the gap between the current model and a proper harmonic representation, the author created an isolated grammar to represent triadic sequences in melodies. The motivation for this was to create a grammar which could identify a single underlying triad for a given note sequence, with hopes to extend the grammar to model sequences of triads and their corresponding transitions.

A triad has three possible notes, and can span any octave. Since every rule must represent relative pitch distance, it would be possible to compute the relative pitch between every note combination of the three notes. To begin, a rule is created that represents the transition from the root note to the third of the the triad, followed by the transition from the third to the fifth of the triad, in pitch classes:

ROOT $\rightarrow$ 3 4

This rule completes the triad, by visiting each note in the triad, starting from whatever note was first played. This by no means covers all the possible triads. Therefore, an abstraction is made to represent moving to the third of the triad. Then a rule is created for the third of the triad moving to the fifth, which will complete the triad:

ROOT $\rightarrow$ 3 THIRD
THIRD $\rightarrow$ 4

This abstraction is extended to label situations in which two of the triadic notes have already been visited as well in order to create a full arpeggiation grammar. The following is a grammar that allows repetition of notes, while ensuring that every note in the triad is visited for every possible combination of rules:

S $\rightarrow$ ROOT | THIRD | FIFTH
ROOT $\rightarrow$ 3 ROOT_THIRD | 7 ROOT_FIFTH
THIRD $\rightarrow$ 4 THIRD_FIFTH | 9 THIRD_ROOT
FIFTH $\rightarrow$ 5 FIFTH_ROOT | 8 FIFTH_THIRD
ROOT_THIRD $\rightarrow$ 4 | 4 FIFTHX | 9 THIRD_ROOT
ROOT_FIFTH $\rightarrow$ 8 | 8 THIRDX | 5 FIFTH_ROOT
THIRD_FIFTH $\rightarrow$ 5 | 5 ROOTX | 8 FIFTH_THIRD
THIRD_ROOT $\rightarrow$ 7 | 7 FIFTHX | 3 ROOT_THIRD
FIFTH_ROOT $\rightarrow$ 3 | 3 THIRDX | 7 ROOT_FIFTH
FIFTH_THIRD $\rightarrow$ 9 | 9 ROOTX | 4 THIRD_FIFTH
ROOTX $\rightarrow$ 3 | 7 | 3 THIRDX | 7 FIFTHX
THIRDX $\rightarrow$ 4 | 9 | 4 FIFTHX | 9 ROOTX
FIFTHX $\rightarrow$ 5 | 8 | 5 ROOTX | 8 THIRDX

This final grammar generates every possible interval combination for ascending arpeggios of a minor triad. The NLTK package provides a module for randomly generating sentences of the language defined by a CFG. This module was used to generate the following example sequences in Table 2 from the triadic grammar just described. The file that contains the CFG sampling code is *generate_arpeggio.py*.

If a melody remains in a single harmonic context (without transitioning to a new underlying chord) it would be possible to extend the arpeggiating grammar above with the rules of embellishment.

Conclusion

This research has performed the induction of a PCFG from a treebank of solutions for the process of melodic reduction. It was shown that, for the most part, adding metric or harmonic information in the data representation improves the efficacy of the resulting probabilistic model, when analyzing the results for the model's ability to reduce melodies in a musically sound way. The source code for this work also allows any researcher to create their own interval representations, and convert the GTTM dataset into a PCFG treebank.

Furthermore, examples of the application of the resulting model for the generation of melodies was also shown. While the compositional value of the melodies is not strong, the combination of the basic building blocks of composition allows it to generate familiar figures.

The lack of harmonic representation is certainly a limitation. If there were a way to identify *which* chord the note belonged to, it would likely help with the grouping of larger phrases in the reduction hierarchy. Furthermore, there is no way to explicitly identify repetition in the melodies with this model. That, too, might be able to assist the model, because if it can identify similar phrases, it could potentially identify the structural notes on which those phrases rely. Each of these improvements would also likely improve the quality of the generated melodies.

The source code for this research is available to the public, and can be found on the author's github account[2].

[2]https://github.com/bigpianist/SupervisedPCFG_MelodicReduction

	ROOT		THIRD		FIFTH	
D.	Intervals	Notes	Intervals	Notes	Intervals	Notes
4	[3, 4]	C, Eb, G	[4, 5]	Eb, G, C	[5, 3]	G, C, Eb
5	[3, 4, 5]	C, Eb, G, C	[4, 5, 3]	Eb, G, C, Eb	[5, 3, 4]	G, C, Eb, G
5	[3, 4, 8]	C, Eb, G, Eb	[4, 5, 7]	Eb, G, C, G	[5, 3, 9]	G, C, Eb, C
5	[3, 9, 7]	C, Eb, C, G	[4, 8, 9]	Eb, G, Eb, C	[5, 7, 8]	G, C, G, Eb
4	[7, 8]	C, G, Eb	[9, 7]	Eb, C, G	[8, 9]	G, Eb, C
5	[7, 8, 4]	C, G, Eb, G	[9, 7, 5]	Eb, C, G, C	[8, 9, 3]	G, Eb, C, Eb
5	[7, 8, 9]	C, G, Eb, C	[9, 7, 8]	Eb, C, G, Eb	[8, 9, 7]	G, Eb, C, G
5	[7, 5, 3]	C, G, C, Eb	[9, 3, 4]	Eb, C, Eb, G	[8, 4, 5]	G, Eb, G, C
10	[7, 8, 9, 7, 5, 3, 9, 7]	C, G, Eb, C, G, C, Eb, C, G	[9, 3, 9, 7, 8, 4, 8, 9]	Eb, C, Eb, C, G, Eb, G, Eb, C	[8, 4, 8, 4, 5, 3, 4, 8]	G, Eb, G, Eb, G, C, Eb, G, Eb

Table 2: Ascending arpeggios generated from the CFG for a minor triad. The "D." column stands for grammar tree "Depth", and is proportional to the number of intervals contained in each grammar string. The grammar was run with the three different starting rules of `ROOT`, `THIRD`, and `FIFTH`. The intervals were then converted to individual notes starting from the root, third, and fifth of the C minor triad, respectively. All trees of depth five or less are included, as well as one randomly-selected grammar string of depth ten.

References

Abdallah, S. A., and Gold, N. E. 2014. Comparing models of symbolic music using probabilistic grammars and probabilistic programming. In *Proceedings of the International Computer Music Conference*, 1524–31.

Abdallah, S. A.; Gold, N. E.; and Marsden, A. 2016. Analysing symbolic music with probabilistic grammars. In Meredith, D., ed., *Computational Music Analysis*. Cham, Switzerland: Springer International. 157–89.

Backus, J. W. 1959. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. In *Proceedings of the International Conference for Information Processing*, 125–31.

Baroni, M., and Jacobini, C. 1975. Analysis and generation of Bach's chorale melodies. In *Proceedings of the International Congress on the Semiotics of Music*, 125–34.

Baroni, M., and Jacoboni, C. 1978. *Proposal for a grammar of melody: The Bach Chorales*. Montreal, Canada: Les Presses de l'Université de Montréal.

Baroni, M.; Brunetti, R.; Callegari, L.; and Jacoboni, C. 1982. A grammar for melody: Relationships between melody and harmony. In Baroni, M., and Callegari, L., eds., *Musical Grammars and Computer Analysis*, 201–18.

Beach, D. 1969. A Schenker bibliography. *Journal of Music Theory* 13(1):2–37.

Chomsky, N. 1956. Three models for the description of language. *Institute of Radio Engineers Transactions on Information Theory* 2:113–24.

Chomsky, N. 1959. On certain formal properties of grammars. *Information and Control* 2(2):137–67.

Gilbert, É., and Conklin, D. 2007. A probabilistic context-free grammar for melodic reduction. In *Proceedings for the International Workshop on Artificial Intelligence and Music, International Joint Conference on Artificial Intelligence*, 83–94.

Hamanaka, M.; Hirata, K.; and Tojo, S. 2007. Implementing "A generative theory of tonal music". *Journal of New Music Research* 35(4):249–77.

Hamanaka, M.; Hirata, K.; and Tojo, S. 2015. σGTTM III: Learning based time-span tree generator based on PCFG. In *Proceedings of the Symposium on Computer Music Multidisciplinary Research*.

Jurafsky, D., and Martin, J. H. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NJ: Prentice Hall, 1st edition.

Kirlin, P. B. 2014. *A probabilistic model of hierarchical music analysis*. Ph.D. thesis, University of Massachusetts Amherst, Amherst, MA.

Lerdahl, F., and Jackendoff, R. 1983. *A generative theory of tonal music*. Cambridge, MA: The MIT Press.

Loper, E., and Bird, S. 2002. NLTK: The natural language toolkit. In *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, volume 1, 63–70.

Marsden, A. 2010. Recognition of variations using automatic Schenkerian reduction. In *Proceedings of the International Conference on Music Information Retrieval*, 501–6.

Roads, C., and Wieneke, P. 1979. Grammars as representations for music. *Computer Music Journal* 3(1):48–55.

Ruwet, N. 1975. Theorie et methodes dans les etudes musicales. *Musique en Jeu* 17:11–36.

Smoliar, S. W. 1976. Music programs: An approach to music theory through computational linguistics. *Journal of Music Theory* 20(1):105–31.