AUTOMATIC MELODIC REDUCTION USING A SUPERVISED PROBABILISTIC CONTEXT-FREE GRAMMAR

Ryan Groves
groves.ryan@gmail.com

ABSTRACT

This research explores a Natural Language Processing technique utilized for the automatic reduction of melodies: the Probabilistic Context-Free Grammar (PCFG). Automatic melodic reduction was previously explored by means of a probabilistic grammar [11] [1]. However, each of these methods used unsupervised learning to estimate the probabilities for the grammar rules, and thus a corpus-based evaluation was not performed. A dataset of analyses using the Generative Theory of Tonal Music (GTTM) exists [13], which contains 300 Western tonal melodies and their corresponding melodic reductions in tree format. In this work, supervised learning is used to train a PCFG for the task of melodic reduction, using the tree analyses provided by the GTTM dataset. The resulting model is evaluated on its ability to create accurate reduction trees, based on a node-by-node comparison with ground-truth trees. Multiple data representations are explored, and example output reductions are shown. Motivations for performing melodic reduction include melodic identification and similarity, efficient storage of melodies, automatic composition, variation matching, and automatic harmonic analysis.

1. 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. The reasons for removing a particular note are, among others, pitch placement, metrical strength, and relationship to the underlying harmony. Because of its complexity, formal theories on melodic reduction that comprehensively define each step required to reduce a piece in its entirety are relatively few.

Composers have long used the rules of ornamentation to elaborate certain notes. In the early 1900s, the music theorist Heinrich Schenker developed a hierarchical theory of music reduction (a comprehensive list of Schenker's publications was assembled by David Beach [7]). Schenker ascribed each note in the musical surface as an elaboration of a representative musical object found in the deeper levels of reduction. The particular categories of ornamentation that were used in his reductive analysis were *neighbor tones*, *passing tones*, *repetitions*, *consonant skips*, and *arpeggiations*. Given a sequence of notes that can be identified as a particular ornamentation, an analyst can remove certain notes in that sequence so that only the more important notes remain.

In the 1980s, another theory of musical reduction was detailed in the GTTM [16]. 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. This differs from Schenkerian analysis, in that every event is related to another single musical event. 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 (see Section 2.0.0.0.2). However, most of these approaches were not data-driven, and those that were data-driven often utilized unsupervised learning rather than supervised learning. A dataset for the music-theoretical analysis of melodies using GTTM has been created in the pursuit of implementing GTTM as a software system [13]. 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 notated using tree structures, which are 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 [11], 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. Using 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 pro-

cess of melodic reduction, using a supervised Probabilisitic Context-Free Grammar (PCFG). By utilizing a ground-truth dataset, it is possible to directly induce a grammar from the solution trees, creating the set of production rules for the grammar and modelling the probabilities for each rule expansion. In fact, this is the first research of its type that seeks to directly induce a grammar for the purpose of melodic reduction. 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 next 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, and also details the different data representations used. Section 6 presents the experiment, including the comparison and evaluation method, and the results of the different tests performed. Section 7 provides some closing remarks.

2. 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.

2.1 Implementing GTTM

While much research has been inspired by GTTM, some research has been done to implement GTTM directly. Fred Lerdahl built upon his own work by implementing a system for assisted composition [17]. Hamanaka et al. [13] presented a system for implementing GTTM. The framework identifies time-span trees automatically from monophonic melodic input, and attained an f-measure of 0.60. Frankland and Cohen isolated the grouping structure theory in GTTM, and tested against the task of melodic segmentation [10].

2.2 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 [20]. Ruwet [21] suggested that a generative grammar would be an excellent model for the creation of a top-down theory of music. Smoliar [22] attempted to decompose musical structure (including melodies) from audio signals with a grammar-based system.

Baroni et al. [4] 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 to analyze and generate melodies in the style of major-mode chorales by Bach [5, 6]. The output of the system would generate the soprano part of the first two phrases of the chorale.

2.3 Probabilistic Grammars

Gilbert and Conklin [11] 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 [1], 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, given in bits per note (bpn). The authors found that the grammar designed by Gilbert and Conklin was the best performer with 2.68 bpn over all the datasets, but one of the Markov model methods had a very similar performance. The same authors also collaborated with Marsden [2] 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 [12]. The authors used the dataset of treebanks that they had previously created [13] 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. 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.

2.4 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 [19]. 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 [15]. Kirlin did not explicitly use a PCFG, however his model was quite similar, and also was a supervised learning method.

3. SUPERVISED LEARNING OF A PCFG

To understand the theoretical framework of the PCFG, it is first useful to give a brief background of formal grammars. Grammars were formalized by Chomsky [8] and extended by himself [9] and Backus et al. [3]. The definition of a formal grammar consists of four parameters, $G = \{N, \Sigma, R, S\}$, which are defined as follows [14]:

N	a set of non-terminal symbols
Σ	a set of terminals (disjoint from N)
R	a set of production rules, each of the form $\alpha \rightarrow \beta$
S	a designated start symbol

Each production rule has a right-hand side, β, that represents the expansion of the term found on the left-hand side, α. 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. Non-terminals are variables that can be expanded (by other rules), while terminals are specific strings, representing elements that are found directly in the sequence (for example, the 'dog' terminal could be one expansion for the *Noun* non-terminal). Given a CFG and an input sequence of terminals, the CFG can *parse* the sequence, creating a hierarchical structure by iteratively finding all applicable rules. Grammars can be ambiguous; there can be multiple valid tree structures for one input sequence.

PCFGs extend the CFG by 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. 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. The parse tree with the maximum probability is the most likely. This process is called disambiguation.

3.1 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. 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.

4. 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 it utilizes 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.

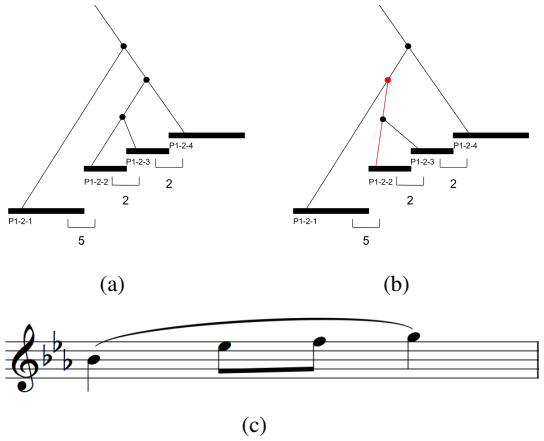

Figure 1: The prolongational tree (a) and the time-span tree (b) for the second four notes in Frédéric Chopin's "Grande Valse Brillante", as well as the score (c). The intervals between notes are notated in number of semitones.

An example helps to detail their differences. Figure 1 shows a particular phrase from one of the melodies in the GTTM dataset: Frédéric Chopin's "Grande Valse Brillante" [13]. The note labelled P1-2-2 is attached to the last note of the melody in the prolongational reduction, because of the passing tone figure in the last 3 notes, whereas the time-span tree connects note P1-2-2 to the first note of the melody, due to its metrical strength and proximity.

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.

5. FORMING THE PCFG

Gilbert and Conklin decided to model the relevant characteristics of the data by hand, by manually creating grammar rules that represented the music composition rules of ornamentation [11]. The melodic embellishment rules included in their grammar were the following: passing tone, neighbor tone, repeated tone, and the escape tone. Additionally, they created a "New" rule which was a kind of catch-all for any interval sequence that could not be described by the other rules. In order for the rules to be applicable at

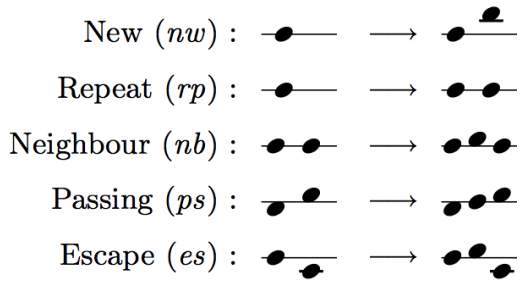

Figure 2: A visualization of a set of melodic embellishment rules, encoded manually into the production rules of a formal grammar [11, 3].

any pitch location, the fundamental unit of data was the interval between two notes, rather than two separate values for each note. The complete ruleset is shown in Figure 2.

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, each rule modelled 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, 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). The "New" rule was no longer needed, since the model learns the rules directly from the training data. This form of one interval expanding into two consecutive intervals for the grammatical rules was adopted for this research.

5.1 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 3. 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

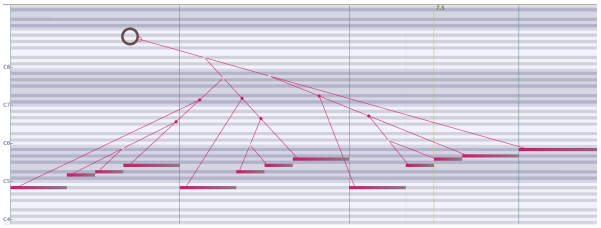

Figure 3: The prolongational reduction tree for half of the first melody in the GTTM dataset, Frédéric Chopin's "Grande Valse Brillante", as displayed in the GTTM visualizer provided by Hamanaka, Hirata, and Tojo [13].

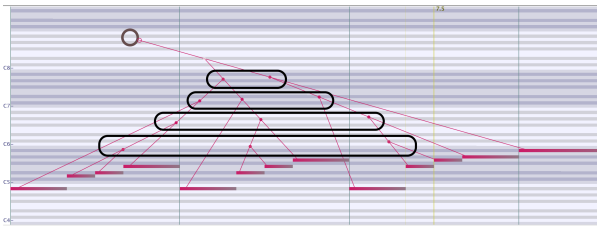

Figure 4: A depiction of the process for converting a tree that uses a note representation to a tree that uses an interval representation, by traversing the tree breadth-wise and relating sets of 3 notes.

the 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. The framework allows for any interval-based representation to be applied. For example, it could be regular pitch intervals, inter-onset interval (IOI), difference in metric prominence, or even representations that consider the notes' relationships to scale and harmony. Figure 4 highlights the breadth-wise traversal process.

The framework was built in Python. It takes a function as input, which allows the user to define unique interval representations. When the function is called during the tree conversion process, the information available for defining the representation consists of the two notes (which contain duration, onset and pitch information), the current key, and the current underlying harmony (if available). The interval encoding that is returned by the function is then used as a node in the resulting tree.

5.2 Training/Induction

The Python-based Natural Language Toolkit (NLTK) was used for the process of PCFG induction [18]. 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 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.

5.3 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 [11]. 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. This metrical hierarchy was also used in GTTM for the metrical structure theory [16].

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 (due to the rule not existing in the training set), 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 change in metric delta (whether the metric level of the subsequent note was higher, the same, or lower than the previous note), 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.

6. 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. Finally, the results of cross-fold evaluation for the PCFG created with each different data representation are shown.

6.1 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 [13, 15]. 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.

6.2 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.

6.3 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

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 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.

It is possible that the lack of data and the subsequent limitation on the complexity of the data representation could be avoided by the use of probabilistic smoothing techniques (to estimate the distributions of those rules that did not exist in the training set) [14, 97]. Indeed, the use of the **Key-Relative Pitch-class** feature as the basis for most of the representations was an attempt to limit the number of resulting rules, and therefore the number of zero-probability rules. This would be an appropriate topic for future experimentation.

A specific example helps to illustrate both the effectiveness and the drawbacks of using the induced PCFG for melodic reduction. Figure 5 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 decisions at the deeper levels of reduction that it chooses incorrect notes as the more important tones.

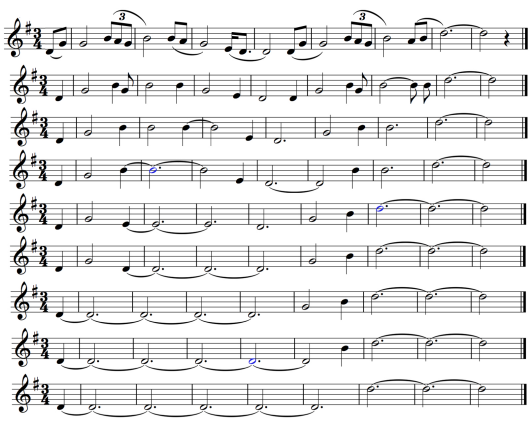

Figure 5: A set of melodies that show the progressive reductions, using the data representation that includes key-relative pitch-class, metric delta and chord tone features.

7. CONCLUSION

This research has performed for the first time 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. A specific example reduction was generated by the best-performing model. There is still much room for improvement, because it seems that the model is more effective at identifying melodic embellishments on the musical surface, and is not able to identify the most important structural notes at deeper layers of the melodic reductions. 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.

There are some specific areas of improvement that might benefit this method. Currently there is no way to identify which chord a note belongs to with the grammar—the harmonic data is simply a boolean that describes whether or not the note is a chord tone. 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. For example, if a group of consecutive notes belong to the same underlying harmony, they could be grouped together, which might allow the PCFG to better identify the more important notes (assuming they fall at the beginning or end of phrases/groups). Beyond that, it would be greatly helpful if the sequences of chords could be considered as well. 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.

The source code for this research is available to the public, and can be found on the author's github account [1] .

[1] http://www.github.com/bigpianist/SupervisedPCFG_MelodicReduction

8. REFERENCES

[1] Samer A. Abdallah and Nicolas E. Gold. Comparing models of symbolic music using probabilistic grammars and probabilistic programming. In *Proceedings of the International Computer Music Conference*, pages 1524–31, Athens, Greece, 2014.

[2] Samer A. Abdallah, Nicolas E. Gold, and Alan Marsden. Analysing symbolic music with probabilistic grammars. In David Meredith, editor, *Computational Music Analysis*, pages 157–89. Springer International, Cham, Switzerland, 2016.

[3] John W. Backus. 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*, pages 125–31, Paris, France, 1959.

[4] Mario Baroni, R. Brunetti, L. Callegari, and C. Jacoboni. A grammar for melody: Relationships between melody and harmony. In Mario Baroni and L Callegari, editors, *Musical Grammars and Computer Analysis*, pages 201–18, Florence, Italy, 1982.

[5] Mario Baroni and C. Jacobini. Analysis and generation of Bach's chorale melodies. In *Proceedings of the International Congress on the Semiotics of Music*, pages 125–34, Belgrade, Yugoslavia, 1975.

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

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

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

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

[10] B. Frankland and Annabel J. Cohen. Parsing of melody: Quantification and testing of the local grouping rules of Lerdahl and Jackendoff's "A generative theory of tonal music". *Music Perception*, 21(4):499–543, 2004.

[11] Édouard. Gilbert and Darrell Conklin. 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*, pages 83–94, Hyderabad, India, 2007.

[12] Masatoshi Hamanaka, K. Hirata, and Satoshi Tojo. σGTTM III: Learning based time-span tree generator based on PCFG. In *Proceedings of the Symposium on Computer Music Multidisciplinary Research*, Plymouth, UK, 2015.

[13] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing "A generative theory of tonal music". *Journal of New Music Research*, 35(4):249–77, 2007.

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

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

[16] Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. The MIT Press, Cambridge, MA, 1983.

[17] Fred Lerdahl and Yves Potard. *La composition assistée par ordinateur*. Rapports de recherche. Institut de Recherche et Coordination Acoustique/Musique, Centre Georges Pompidou, Paris, France, 1986.

[18] Edward Loper and Steven Bird. 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, pages 63–70, Stroudsburg, PA, 2002.

[19] Alan Marsden. Recognition of variations using automatic Schenkerian reduction. In *Proceedings of the International Conference on Music Information Retrieval*, pages 501–6, Utrecht, Netherlands, August 9-13 2010.

[20] Christopher Roads and Paul Wieneke. Grammars as representations for music. *Computer Music Journal*, 3(1):48–55, March 1979.

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

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