Recursive Generation of Rhythmic Structures with PTGGs

Brian Heim
Yale University
brian.heim@yale.edu

Donya Quick
Stevens Institute of Technology
dquick@stevens.edu

ABSTRACT

We present an approach for generating passages of rhythm via the use of a probabilistic temporal graph grammar with rule sets that operate via recursive subdivision. This project is motivated by three factors: (1) the recognition that the literature on computer-generated rhythm is generally not as robust as those for melody and harmony; (2) the creation of more tools for computer-assisted composition; and (3) the hope that such an approach, having already produced compelling results for some styles of music, may provide insight into the structure of rhythms found in existing repertoires.

1. INTRODUCTION

There has been extensive work in the areas of analyzing and generating pitch-related features in music such as melody and harmony. However, rhythmic structure has received substantially less attention, particularly in the area of automated composition. Typically, generated rhythms are either very simple, such as those found in Bach chorales, or they struggle to retain a sense of meter.

We present a grammatical approach to generating diverse rhythmic patterns in multiple time signatures while preserving hypermetrical structures. Our approach also allows the generation of repeated patterns at multiple levels of granularity, an extremely important feature of rhythm in music. Finally, the probabilistic grammars described here also have the potential for rule probabilities to be derived from a corpus of real musical examples.

As both authors are familiar with the Common Practice era of Western classical music, the model and examples presented here derive significantly from that corpus. A partial goal of the initial implementation of this approach was to aid in computer-assisted composition. Particular design choices were made in order to make idioms common in contemporary Western classical music easily accessible; the result of this may be seen in the "complex" example below (Figure 3). In Future Work, we also discuss the possibility of adapting the implementation for features common to other genres of music.

1.1 Related Work

One of the most common algorithms used in computer music analysis and generation is the Markov chain [1, 2, 3, 4]. However, Markov chains are fundamentally limited in the temporal scope of features that they can capture, which is problematic for modeling metrical structure and rhythmic patterns—for which temporal context and repetition are extremely important. As the order of the chain increases, more complex features can be modeled, but observations of states become sparse and the number of states increases exponentially. Even approaches such as variable-length Markov chains [5, 6] and the extensions noted above do not entirely overcome these problems.

Some extensions to Markov chains exist that can overcome some of the problems of preserving metrical structure observed with more traditional approaches. Roy and Pachet model meter in melodic generation [7] using an extension known as Markov constraints [8] to achieve properties such as properly filled measures. This method involves enumeration of possible outcomes for metrical constraints, although the complexity is mitigated to some extent by a filtering step to avoid a truly brute force search.

Work such as that of Lerdahl and Jackendoff [9], as well as the work of Temperley [10], presents a statistical view of metrical structure, where metrical positions are assigned a probability distribution and downbeats are weighted more heavily. While these models are useful analytically and can be used to generate new rhythmic patterns, as with Markov chains, there is no support for modeling repetition, and relatively common, syncopated rhythmic patterns are unlikely due to having little downbeat content.

Grammars fundamentally have more expressive power than Markov chains, with some categories of grammar supporting long-term constraints and even repetition. Because of this, we feel that grammatical approaches are better suited to modeling rhythm and metrical structure than purely Markovian methods.

Keller and Morrison present a grammatical model for jazz solos that includes a rhythmic component[11]. This grammar is context-free and generates rhythmic structure independently of melodic structure. The rhythmic portion of the grammar is fundamentally based on dividing times in half. Although some degree of syncopation is supported, the strategy of halving durations through much of the generation fundamentally limits it to duple meters. The grammars we introduce in later sections of this paper use a tiered, multi-rule-set approach that supports many other time signatures.

1.2 Probabilistic Temporal Graph Grammars

A grammar is defined as a 4-tuple: $G = (N, T, R, S)$ where the alphabet $N \cup T$ comprises nonterminals, N, and terminals, T. R is a set of rules of the form $x \in N \rightarrow (N \cup T)^*$ and $S \in N$ is the starting sym-

bol. Terminals are symbols that only produce themselves, i.e. cannot be further altered, while nonterminals may produce any combination of terminals and nonterminals. The condition that production rules operate on individual symbols without knowledge of the surrounding symbols means that this definition also describes a Context-Free Grammar (CFG). Probabilistic context-free grammars (PCFGs) have more than one rule with the same left-hand side. Rules sharing the same left-hand side in a PCFG must have associated probabilities that sum to 1.0.

Kulitta features a category of grammars called Probabilistic Temporal Graph Grammars[12, 13], or PTGGs. PTGGs are essentially parameterized PCFG, where production rules are functions on parameters. Allowable functions on symbol parameters include simple conditional logic and let-in statements for variable creation and instantiation. These let-in statements allow the generation of repetitions, which are an important feature of rhythm in music.

The generative algorithm used with PTGGs is similar to that of L-Systems: at each generative iteration of the grammar, starting with the single-symbol sequence of $X = S$, the appropriate rule $(x \to y) \in R$ is applied to each nonterminal symbol from left to right to produce a new sequence X', which may contain more symbols than X. The sequence length is a nondecreasing function of the number of iterations.

Rules in PTGGs have the following format:

$$\begin{aligned} x^p \to \quad & f_1(p) \\ & \mid \textbf{if } cond(p) \textbf{ then } f_1(p) \textbf{ else } f_2(p) \\ & \mid \textbf{let } x = f_1(p) \textbf{ in } \{x \mid f_i(p)\}^+ \end{aligned}$$

where each $f_i(p)$ indicates some function from parameter values to sequences of terminals and nonterminals and $cond(p)$ is a conditional test on the left-hand-side symbol's parameter (such as $p < 2$).

2. A PTGG FOR RHYTHM

In most PTGGs used by Kulitta, the alphabet consists of Roman numeral chord symbols parameterized by duration, and rules are *duration preserving*, meaning that the right-hand side symbols divide up the duration of the left-hand symbol, typically by a power of two. This division always results in a string that has equal duration to the original symbol. A rule such as $I^t \to I^{t/2}V^{t/4}I^{t/4}$ indicates that a I-chord may be replaced by a I-V-I progression where the chord durations are $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{4}$ the durations of the original chord. This implies a general approach of beginning with a long duration and iteratively dividing it into progressively smaller durations; since one would typically like to have control over the exact duration of a generated composition, this is a reasonable approach.

The grammar proposed here is based on the observation that many theories of Western counterpoint, including those of Fux, Schoenberg, and Schenker [14, 15], treat rhythm as a hierarchy of nested structures, discuss metrical patterns such as anapest (short-short-long) in the same light regardless of the hierarchical level on which they occur, and regard small note values as elaborations or ornamentations of a less complex metrical pattern.

We also introduce the notion of *production phases* into the PTGG framework to account for the need for different rules to take place at different granularities in the generative process.

Durations are treated as both terminals and nonterminals of an infinite parametrized alphabet of duration types in a four-phase generation algorithm process, each step of which subdivides an initial duration into smaller and smaller units.

Throughout the following sections, we use 1.0 to represent the duration of a whole note in 4/4.

2.1 Nonterminals

The nonterminals of the grammar are $N = \{Beat, Dotted, QuarterDotted, Measures\} \cup TS$ where TS is a set of *time signature symbols*.

The *Beat* symbol represents a note with duration $2^n/m$ where n and m are integers with $n \geq 0$, $m > 0$. The *Dotted* symbol represents a duration analogous to a dotted note; it lasts 1.5 times its undotted value. Similarly, a *QuarterDotted* lasts for 1.25 times its undotted value. Although the notion of a "quarter dotted" note has no counterpart in common practice notation (although some composers such as George Crumb have invented their own symbols for it), a duration of $\frac{5}{8}$ is found frequently in situations where, for example, a quarter note is tied to the first of a series of sixteenth notes. Because the grammar has no way to represent ties, this symbol is included for robustness.

One could argue that, for the sake of completion, symbols for $\frac{7}{8}$, $\frac{9}{8}$, and so on ought also to be defined. We have chosen to forego defining these symbols because (1) their durations occur less frequently than $\frac{3}{8}$ and $\frac{5}{8}$, (2) we are still able to express a wide variety of rhythms without them, (3) there are other, more elegant routes to the same solution (see Future Work).

Measures represents a nonzero length of consecutive measures; each measure has nonzero duration but has not yet been assigned a time signature. Finally, TS is the set of time signatures that may appear in the final output; symbols in TS are named for the meters they represent: *FourFour*, *ThreeFour*, *SixEight*, and so on.

2.2 Terminals

The set of terminals consists of one element: $T = \{Short\}$. *Short* stands for a *Beat* which has been subdivided to the point where its duration is less than or equal to $\frac{1}{2^n}$, where n is an arbitrary integer constant. Intuitively, a very short note is prevented from being further subdivided. This behavior is designed to mimic the limits of human performers rhythmic articulation.

The notations B^d, S^d, D^d, Q^d will be used hereafter for $Beat$, $Short$, $Dotted$, and $QuarterDotted$ symbols of duration d respectively. We also use the notation M^l for a $Measures$ symbol with length l.

2.3 Production Phases

Production occurs in four phases of rule sets. A sentence is considered well-formed if and only if it passes through all four phases. These phases correspond to the generation of (1) phrase structure, (2) time signatures, (3) intra-measure beat patterns, and (4) recursive rhythmic subdivision. Having four separate rule sets, each with a well-defined pur-

pose, allows us a great deal of flexibility without forcing us to over-specify.

Phase 1 begins with the string $\alpha = M^l$ where $l = 2^a$, $a \geq 0$, and halts with the string $\beta = \{M^1\}^*, |\beta| = l$.

Phase 2 generates γ from β by replacing each M^1 in β with a symbol $ts \in TS$, where TS is a nonempty set of time signature symbols.

Phase 3 generates δ from γ by replacing each ts in γ with a sequence of symbols from $\{Beat, Short, Dotted, QuarterDotted\}$ such that each sequence's duration sums to the duration of the time signature it replaces. For instance, the sequence $[B^{\frac{1}{2}} D^{\frac{3}{8}} B^{\frac{1}{8}}]$ may be validly generated from *FourFour* through the application of the third and then second rules in Table 2, while the sequence $[B^{\frac{1}{2}} D^{\frac{3}{4}}]$, however, may not be generated from *FourFour*.

Phase 4 generates the output string by repeatedly subdividing δ. Rules in this phase operate within the duration-parametrized set $\{Beat, Short, Dotted, QuarterDotted\}$. In order to preserve the duration of δ, the durations of the right-hand side of a rule must sum to the duration of the left-hand side.

2.4 Production Rules

Production rules in Phase 1 are functions from measures to measure groups, $M \rightarrow M^+$, and in Phase 2 from beats to beat groups, $B \rightarrow \{B, D, S\}^+$. The strict definition of PFCG says that a single rule must produce a single string of symbols, as in the case of $B \rightarrow BB$ and $B \rightarrow SS$. Yet *Short*, as noted above, is only produced when its parameter meets certain conditions. Since the decision to output B, D or S can be made solely on the basis of the parameter value, we define rules as functions:

$B^a \rightarrow X \in (N \cup T)+$ where
$\forall x^d \in X : 0 < d \leq 1, \sum_d = 1$

and then use the resulting parameters $[d_1, d_2, \ldots, d_p] = aX$ to label the produced symbols. In words, this means that the right-hand side is a list of fractions, summing to 1, that describe how the left-hand side is to be subdivided. This allows us to define patterns of subdivision clearly and succinctly. For example, let two Phase 4 rules be $B \rightarrow [\frac{3}{4}, \frac{1}{4}]$ and $B \rightarrow [\frac{1}{2}, \frac{1}{4}, \frac{1}{4}]$. Here are two iterations of production using these rules and a *Short* threshold of $\frac{1}{16}$:

Rule	String
(start)	B^1
$B \rightarrow [\frac{3}{4}, \frac{1}{4}]$	$D^{\frac{3}{4}} B^{\frac{1}{4}}$
$B \rightarrow [\frac{1}{2}, \frac{1}{4}, \frac{1}{4}]$	$D^{\frac{3}{4}} B^{\frac{1}{8}} S^{\frac{1}{16}} S^{\frac{1}{16}}$

3. IMPLEMENTATION

We use Haskell-based PTGG generation framework from Kulitta and the MIDI export capabilities offered by the Euterpea library [16] to allow for elegant rule definition and parsing. Below are some of the key features of this implementation.

3.1 Four-Phase Generation Algorithm

Assuming the rule sets have already been defined, the complete generation algorithm requires four parameters: (1) the total length $l = 2^n$, $n > 0$, of the passage in measures; (2) the shortest allowable duration $1/2^s$ expressed as s; (3) the number of Phase 4 iterations to perform; and (4) a random seed.

0.33	$M \rightarrow$ **let** $x = M^{\frac{l}{2}}$ **in** $x\ x$
0.67	$M \rightarrow M^{\frac{l}{2}} M^{\frac{l}{2}}$
1.00	$M \rightarrow FourFour$
1.00	$FourFour \rightarrow B^1$

Table 1. Rules, along with their corresponding probabilities, used in Phases 1, 2 & 3 to generate Figure 1

As stated above, generation begins with the string $\alpha = M^l$. Phase 1, phrase structure generation, uses only two rules: $M^l \rightarrow M^{\frac{l}{2}} M^{\frac{l}{2}}$ and **let**$x = M^{\frac{d}{2}}$**in**xx. Both rules subdivide the group of measures into two equal halves, but one does so through variable instantiation. Since l is a power of two, Phase 1 halts in $log_2 l$ iterations.

In Phase 2, time signature assignment, each measure is assigned a time signature from the set TS. Since most pieces of common practice music are in a single time signature, this step will typically only consist of one rule $M \rightarrow ts, \{ts\} = TS$.

In Phase 3, beat pattern assignment, each time signature symbol is replaced with a sequence of duration symbols that sum to the duration of that meter. An example Phase 3 rule might be $FourFour \rightarrow B^{\frac{1}{2}} B^{\frac{1}{2}}$, which divides a 4/4 measure into two independent halves. This Phase may also be trivial for generating common practice passages, but for complex or irregular time signatures such as 7/8, it is helpful to explicitly define the abstract beat pattern as $2 + 2 + 3$, or $3 + 2 + 2$, or even a mix of both.

Finally, Phase 4 replaces each duration symbol with a sequence of durations that sum to the original symbol's duration. The rules in Phase 4 are applied for a predetermined number of iterations. Each Phase 4 rule is actually a conditional predicated on the parameter of the left-hand side: the algorithm first determines if applying the subdivision would produce a duration shorter than the shortest allowed duration. If so, the symbol is left unchanged; otherwise, subdivision occurs, replacing symbols with B, D, or S as described in Production Rules.

As a last step, a single measure consisting of a whole note or its equivalent in the relevant time signature is appended to the resulting string, in order to give a sense of finality.

4. EXAMPLES

We now present three example passages generated from this grammar. The first two are in 4/4 and 3/4 time respectively, with rule sets chosen freely in a rough attempt to imitate the repertoire of the Classical period of Western classical music. The third example is in a mix of irregular meters and demonstrates the ease of extending the grammar to a more contemporary style of rhythm, somewhere between the tuplets of Elliott Carter and Karlheinz Stockhausen and the dense rhythmic variety of Brian Ferneyhough. The purpose of these examples is primarily to give a sense of the variety and complexity possible under this approach, rather than to directly imitate any particular composer, school, or period.

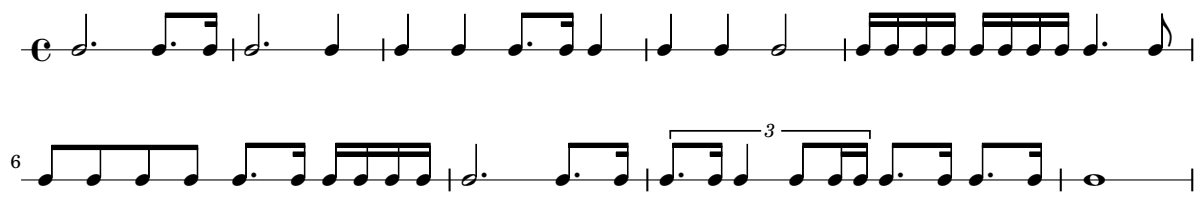

Figure 1. A generated passage of eight measures of 4/4, with an extra whole note measure added for finality.

Figure 2. A generated passage of 3/4. Note the repetition of measures 1 and 2, a result of variable instantiation during Phase 1 of generation.

0.30	$B \rightarrow B^1$
0.20	$B \rightarrow D^{\frac{3}{4}} B^{\frac{1}{4}}$
0.15	$B \rightarrow B^{\frac{1}{2}} B^{\frac{1}{2}}$
0.10	$B \rightarrow B^{\frac{1}{2}} B^{\frac{1}{4}} B^{\frac{1}{4}}$
0.05	$B \rightarrow B^{\frac{1}{4}} B^{\frac{1}{2}} B^{\frac{1}{4}}$
0.05	$B \rightarrow B^{\frac{1}{3}} B^{\frac{1}{3}} B^{\frac{1}{3}}$
0.15	$B \rightarrow$ **let** $x = B^{\frac{1}{2}}$ **in** $x\ x$
0.15	$B \rightarrow$ **let** $x = B^{\frac{1}{4}}$ **in** $B^{\frac{1}{2}}\ x\ x$
1.00	$D \rightarrow D^1$

Table 2. Phase 4 rule set used to generate Figure 1.

1.00	$M \rightarrow ThreeFour$
0.33	$ThreeFour \rightarrow B^{\frac{1}{4}} B^{\frac{1}{4}} B^{\frac{1}{4}}$
0.33	$ThreeFour \rightarrow$ **let** $x = B^{\frac{1}{4}}$ **in** $B^{\frac{1}{4}} x\ x$
0.33	$ThreeFour \rightarrow$ **let** $x = B^{\frac{1}{4}}$ **in** $x\ x\ B^{\frac{1}{4}}$

Table 3. Rules used for Phases 2 and 3 to generate Figure 2. Rule sets for other phases are omitted here for the sake of brevity.

Figure 1 shows an eight-measure, 4/4 passage generated with the rule sets in Tables 1 and 2. The rules for the first three phases are trivial: after subdividing the initial M^8 symbol into a string of eight M^1, each is converted to $FourFour$, and then to B^1, a whole note. The nine rules in Table 2, which were applied for four iterations, did most of the work. From this small set of rules, we are able to generate a highly varied set of rhythms. For instance, $B \rightarrow D^{\frac{3}{4}} B^{\frac{1}{4}}$ has been applied to four distinct durations: a whole note (mm. 1-2), a quarter note (mm. 1, 3, 6-8), a half note (m. 5), and a triplet quarter note (m. 8). Although no rule directly subdivides a beat into four or eight notes of the same duration, phenomena like the eight consecutive sixteenth notes in m. 5 are made more probable through the use of let-in rules.

The quarter-note triplet in m. 8 of Figure 1 reveals one drawback of this approach: some subdivision rules are more idiomatic than others on larger metrical scales. Because we subdivide with the same rule set each time, each rule has the same likelihood of being applied to a quarter note as it does to a whole note. This presents a difficulty: quintuplet subdivisions, though rare, are sometimes encountered in common practice era music as quick ornamental groupings. Including a rule to generate quintuplets would mean accepting the possibility of a whole note becoming a quarter note quintuplet as well.

0.40	$M \rightarrow FifteenSixteen$
0.20	$M \rightarrow ThreeFour$
0.20	$M \rightarrow FiveEight$
0.20	$M \rightarrow SevenEight$
1.00	$ThreeFour \rightarrow B^{\frac{1}{4}} B^{\frac{1}{4}} B^{\frac{1}{4}}$
1.00	$FiveEight \rightarrow B^{\frac{3}{8}} B^{\frac{1}{4}}$
1.00	$FifteenSixteen \rightarrow B^{\frac{1}{4}} B^{\frac{1}{4}} B^{\frac{1}{4}} B^{\frac{3}{16}}$
1.00	$FifteenSixteen \rightarrow B^{\frac{5}{16}} B^{\frac{5}{16}} B^{\frac{5}{16}}$
1.00	$SevenEight \rightarrow B^{\frac{1}{4}} B^{\frac{1}{4}} B^{\frac{3}{8}}$

Table 4. Rule used for Phases 2 and 3 to generate figure **??**. The rule set for Phase 4 has been omitted for the sake of brevity.

This problem could be partially addressed by adding rules to Phase 3 to ensure that a $FourFour$ symbol expands into sensible, larger subdivisions, such as a group of four quarter notes. However, in order to retain the same degree of flexibility, we would have to include many such rules in Phase 3, duplicating the "acceptable" ways in which a whole note could decompose under the Phase 4 rule set. This is hardly a convincing solution, since we would also be dispensing entirely with the possibility of generating an undivided whole note in the final string. We acknowledge that the present solution–setting the probabilities of rules such as the triplet subdivision to be very low–is the best we can do without modifying the implementation (see Future Work).

In Figure 2, we see a 3/4 passage generated by the same grammar. By simply altering a few rules in the early phases of generation, we were able to produce rhythm in a completely different time signature using the same grammar and implementation. Instead of transforming $Measures$ into $FourFour$ symbols, we transform them to $ThreeFour$ and then provide a set of possibilities for decomposing a $ThreeFour$ measure into a sequence of $Beats$. This justifies our earlier separation of the concept of phrase structure from time signature, and time signature from subdivisions of smaller beats. The rules for Phases 2 and 3 are shown in Table 3; Phases 1 and 4 were unchanged, although the Phase 4 rule set was only applied once.

The previous example diversified the rule set of Phase 3,

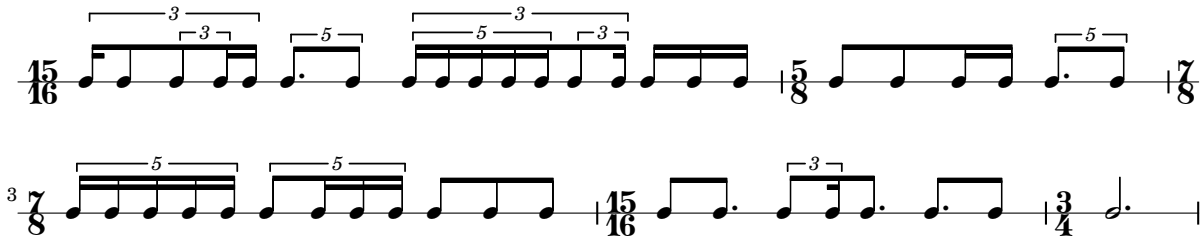

Figure 3. A brief passage demonstrating the ability of this framework to easily generate rhythms in multiple time signatures with complex beat subdivisions. A partial rule set is given in Table 4

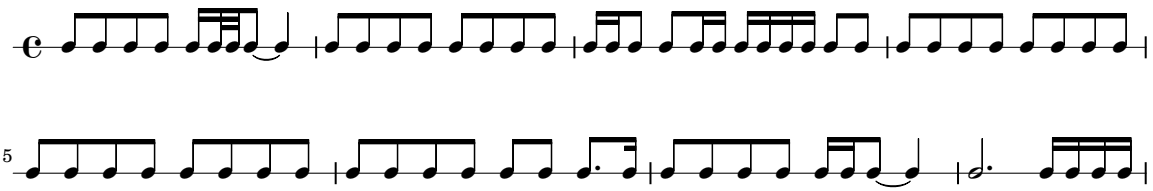

Figure 4. Examples of rhythms generated from the grammar in Table 5, the probabilities for which were derived from a small number of Bach cantata excerpts.

beat pattern generation. What if we wanted to also diversify Phase 2, time signature assignment? Figure 3 shows an example of this grammar in which an abstract measure is allowed to transform into any one of four time signatures, each of which further transforms into one of potentially multiple beat patterns (Table 4). Considering m. 1 of Figure 3, we note that this grammar, by virtue of its ratio-based representation of duration values, gracefully produces nested tuplets of arbitrary denominator.

In a passage like Figure 2, the frequent downbeats and lack of syncopation could quickly become tiresome in a longer passage of 16 or 32 measures. The current model has no way to address this potential for monotony: no matter how a duration is subdivided, the resulting sequence always keeps the initial impulse the same. There are a number of modifications that could be made to the grammar to address this shortcoming (see Future Work).

5. LEARNING PRODUCTION PROBABILITIES

An algorithm for deriving production probabilities from a training corpus has been described for probabilistic temporal graph grammars [17], and is part of the Kulitta framework. This learning method is based on the inside-outside algorithm[18], which, in turn, is a tree generalization of the more commonly used forward-backward algorithm for hidden Markov models.

Kulitta's production probability learning-strategy is oracle-based, allowing PTGG for which the oracle can be defined concretely to have production probabilities estimated from real data. This strategy also supports PTGGs with rules that feature let-in expressions and conditional statements. The oracle must be able to identify candidate parent symbols that may have produced mixed sequences of terminals and nonterminals.

For the PTGGs described in this paper, the learning oracle is simple to define because all of the rules are *duration preserving*. A rule can be tested against a sequence of symbols by normalizing the durations of the input sequence to sum to 1.0 and comparing against symbol parameters in the rule. For example, the sequence B^2B^2, which normalizes to $B \rightarrow B^{\frac{1}{2}}B^{\frac{1}{2}}$ can clearly be produced by the rule $B \rightarrow B^{\frac{1}{2}}B^{\frac{1}{2}}$, but B^1B^2 cannot.

0.851	$B \rightarrow B^{\frac{1}{2}}B^{\frac{1}{2}}$
0.004	$B \rightarrow B^{\frac{1}{2}}B^{\frac{1}{4}}B^{\frac{1}{4}}$
0.030	$B \rightarrow B^{\frac{1}{4}}B^{\frac{1}{2}}B^{\frac{1}{4}}$
0.002	$B \rightarrow B^{\frac{1}{4}}B^{\frac{1}{4}}B^{\frac{1}{2}}$
0.071	$B \rightarrow D^{\frac{3}{4}}B^{\frac{1}{4}}$
0.010	$B \rightarrow B^{\frac{1}{4}}D^{\frac{3}{4}}$
0.010	$B \rightarrow B^{\frac{1}{4}}B^{\frac{1}{4}}B^{\frac{1}{4}}B^{\frac{1}{4}}$
0.010	$B \rightarrow B^{\frac{1}{3}}B^{\frac{1}{3}}B^{\frac{1}{3}}$

Table 5. Rule set with probabilities derived from training on examples of rhythms from Bach cantatas.

As a proof of concept, we tested a reduced rule set using a small collection of rhythmic patterns from Bach cantatas. The production probabilities derived from this experiment are shown in Table 5, and example rhythms generated from the learned model are shown in Figure 4.

6. CONCLUSION AND FUTURE WORK

We have presented an application of PTGGs to the topic of automated composition along the sole parameter of rhythm, including the implementation of several example grammars. These grammars support multiple time signatures and complex rhythmic patterns while still retaining a strong sense of meter in generated sequences due to the use of a tiered approach to generating sequences that addresses hypermetrical structure.

However, there are still rhythmic features missing from our models. One clear area for expansion would be the introduction of rests and ties, which would give access to syncopation, an important feature in many genres of music, especially jazz. These extensions would motivate the introduction of more alphabet symbols: a Tied symbol, pro-

duced from a Beat or Short, could be interpreted as tied over from the previous note during playback. A Rest symbol could be similarly used to eliminate some articulated notes from the interpreted string.

Other concepts missing from the current alphabet are accents, embellishments and other ornamentations. Accents in particular are important to the perception of meter. Embellishments/ornaments could add short note values before or after more metrically significant beats, and in so doing establish an additional level of metrical hierarchy. A separate way to approach this would be by allowing existing let-in structures in the string to be resolved during or between iterations, rather than at the end of the entire cycle. This would preserve high-level symmetry while allowing for small-scale variations.

Another direction to explore is context sensitivity. In some cases, it might be helpful to know certain contextual details about a symbol such as: (1) its position within the measure, which typically determines its emphasis (downbeat vs. upbeat), (2) its position within the overall duration of the passage, and (3) whether the passage is increasing or decreasing in rhythmic activity. Otherwise, it is not possible to account for what Schoenberg termed the tendency of the smallest notes[15]—a phenomenon wherein shorter durations are used more frequently toward the end of a passage or phrase, without adding other constraints to the rule set.

These variations and extensions could also be useful for interfacing our grammar with other automated composition algorithms that produce melodies, harmonies, and counterpoint, either by using its output as a blueprint or by including it in a composition by committee approach. In that case, it would be important to embed as much contextual and functional knowledge as possible into the alphabet and the symbol parameters so that other grammars and algorithms can make informed decisions. For example, the knowledge that a downbeat is tied over or that two halves of a phrase have some rhythmic symmetry could be interpreted by a harmonic composition algorithm to produce more contextually aware chord sequences.

7. REFERENCES

[1] P. Chordia, A. Sastry, and S. Senturk, "Predictive Tabla Modeling using Variable-length Markov and Hidden Markov Models," *Journal of New Music Research*, vol. 40, no. 2, pp. 105–118, 2011.

[2] J. Gillick, K. Tang, and R. M. Keller, "Learning Jazz Grammars," in *Proceedings of the Sound and Music Computing Conference*, 2009, pp. 125–130.

[3] L. Yi and J. Goldsmith, "Automatic Generation of Four-Part Harmony," in *UAI Applications Workshop*, 2007.

[4] F. Pachet, "The Continuator: Musical Interaction With Style," in *Proceedings of the International Computer Music Conference*, 2002, pp. 2011–218.

[5] D. Ron, Y. Singer, and S. Tishby, "The Power of Amnesia: learning Probabilistic Automata with Variable Memory Length," *Machine Learning*, vol. 25, pp. 117–149, 1996.

[6] P. Bühlmann and A. J. Wyner, "Varable Length Markov Chains," *The Annals of Statistics*, vol. 27, no. 2, pp. 480–513, 1999.

[7] P. Roy and F. Pachet, "Enforcing Meter in Finite-Length Markov Sequences," 2013.

[8] F. Pachet and P. Roy, "Markov constraints: steerable generation of Markov sequences," *Constraints*, vol. 16, no. 2, pp. 148–172, 2011.

[9] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music*. The MIT Press, 1996.

[10] D. Temperley, "Modeling Common-Practice Rhythm," *Music Perception*, vol. 27, no. 5, pp. 335–376, 2010.

[11] R. M. Keller and D. R. Morrison, "A Grammatical Approach to Automatic Improvisation," in *Sound and Music Computing Conference*, 2007, pp. 330–337.

[12] D. Quick and P. Hudak, "Grammar-Based Automated Music Composition in Haskell," in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling, and design*, 2013, pp. 59–70.

[13] D. Quick, "Kulitta: a Framework for Automated Music Composition," Ph.D. dissertation, 2014.

[14] J. J. Fux, A. Mann, and J. Edmunds, *Gradus ad Parnassum: The Study of Counterpoint*. WW Norton & Company, 1965.

[15] A. Schoenberg, *Preliminary exercises in counterpoint*, L. Stein, Ed. Faber & Faber, 1963.

[16] P. Hudak *et al.* (2014) Euterpea. [Online]. Available: http://hackage.haskell.org/package/Euterpea

[17] D. Quick, "Learning production probabilities for musical grammars," *Journal of New Music Research*, vol. 45, no. 4, pp. 295–313, 2016.

[18] K. Lari and S. Young, "The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm," *Computer Speech ad Language*, vol. 4, pp. 35–56, 1990.