Recursive Generation of Rhythmic Structures with PTGGs

Brian Heim
Affiliation1
author1@unt.edu

Donya Quick
Affiliation2
author2@unt.edu

ABSTRACT

[Will need to be tweaked at the end to reflect any additions, but the original intro seems like a good fit here at the moment]

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) my desire as a composer of avant-garde classical music to have 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

[To-Do]

1.1 Related Work

[Cite some rhythmic modeling papers here]

[Specific things to cite: Pachet's Markov chains, Lerdahl and Jackendoff's GTTM, Temperley's probabilistic models]

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 $x \in N \to y^*$ where x is a nonterminal symbol and y^* is any string of symbols in $N \cup T$. $S \in N$ is the starting symbol. Terminals are symbols that only produce themselves, i.e. cannot be further altered, while nonterminals may produce any combination of terminals and nonterminals. The generative algorithm is similar to that of L-Systems: at each generative iteration of the grammar, starting with the string $X = S$, the appropriate rule $(n \to y) \in R$ is applied to each nonterminal symbol left to right to produce a new string X'. This process of iteration is called expansion, because the string length is a nondecreasing function of the number of iterations. 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).

A probabilistic context-free grammar (PCFG) is a CFG with a rule set that may contain multiple production rules per nonterminal. These grammars were introduced by the automated composition system, Kulitta, as a means to simultaneously generate harmonic and rhythmic structures [**?**]. Rules take the tuple form $r = (\pi, x \to y^*)$ where $n \in [0, 1]$ is the probability of rule r being applied to an instance of symbol x. We denote the probability of rule r as π_r. There is an added constraint that all probabilities for a given nonterminal sum to 1. As noted by Paul Hudak and Donya Quick [-2], PCFGs are a reasonable approach to the problem of automated music composition, as they are efficient, easily defined, and take into account the fact that probabilistic harmonic behavior has been observed in significant repertories of classical music [8, 9].

[EXTREMELY BRIEF PTGG MATH DEFINITION - BORROW FROM ICMC2015]

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 [**?**], treat rhythm as a hierarchy of nested structures, discuss metrical patterns such as anapest (short-short-long) in the same light regardless of the level of duration on which they occur, and regard small note values as elaborations or ornamentations of a less complex metrical pattern. For the sake of focus the grammar is defined only for measures of $4/4$ meter, since many counterpoint treatises begin inand sometimes never leavethis meter. 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 two-step generative process that first divides a raw duration, equal to a power of two, into a sequence of measures;

second, these measures are transformed into whole note durations and subdivided further. Throughout the following sections, I use the value 1.0 to represent the duration of a whole note, which is also the duration of a measure.

2.1 Nonterminals

The nonterminals of the grammar are $N = Measure,\ Beat$. The *Beat* symbol represents a note with duration $2^n/m$ where n and m are integers with $n \geq 0$, $m \geq 1$; *Measure* is the term used for a raw duration whose value is a positive integer. Measures only appear in the first phase of production and are replaced with Beats of duration 1.0 in the second phase. The reason for this distinction is twofold: first, it was desired that the second phase of production begin with a series of 1.0 durations; second, it allows the frequency of variable instantiation at the hypermetrical level to be controlled independently of that at the metrical level.

2.2 Terminals

The terminals used are $T = Dotted,\ Short$. The *Dotted* symbol represents a duration that, like a dotted note, lasts for 1.5 times the value of its undotted value. It was chosen to be a terminal for convenience since any subdivision of a dotted value can also be articulated as a more complex subdivision of a longer duration. Short stands for a Beat which has been subdivided to the point where its duration is less than or equal to $1/2^n$, where n is an arbitrary and predetermined integer constant. This behavior is designed to mimic the limits of human performers rhythmic articulation. The notations Md, Bd, Sd, Dd will be used hereafter for Measures, Beats, Shorts, and Dotteds of duration d respectively.

2.3 Production Phases

Production occurs in two phases of rulesets, and a sentence is not considered well-formed until passing through both phases. Phase 1 begins with the string Md where$d = 2^a, a \in N$, and ends with the string $A \in K$. Letting A_V be the set of all Md in A and all As variables, Phase 1 only halts when $\forall M^d \in A_V, d = 1$. Phase 2 begins with the string B obtained by replacing all $M^d \in A_V$ with B^d, and halts after a predeterminsed number of iterations. At each iteration of both phases, the sum, D, of durations in the string remains constant.

2.4 Sentential Forms

[Needs work - may be worth cutting]

The set K of sentential forms of this PTGG is

$k \in K ::= M \mid kk \mid let\ x = k\ in\ k$

for the first phase and

$l \in L ::= B \mid (S \mid Dl \mid lD \mid ll) \mid let\ y = l\ in\ l$

for the second, where x and y are variable names.

2.5 Production Rules

Production rules in Phase 1 are functions from measures to measure groups, $M \rightarrow K$, and in Phase 2 from beats to beat groups, $B \rightarrow L$. 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, in 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:

$B^a \rightarrow \mathbb{X} \in \mathbb{Q}^p$ where

$p > 0, \forall x \in X : 0 < x \leq 1, \sum_{x \in X} x = 1$

and then use the resulting parameters $[d_1, d_2, , d_p] = aX$ to label the produced symbols. This allows me to define patterns of subdivision clearly and succinctly. For example, let two Phase 2 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
$B \rightarrow [\frac{3}{4}, \frac{1}{4}]$	$D^{3/4}B^{1/4}$
$B \rightarrow [\frac{1}{2}, \frac{1}{4}, \frac{1}{4}]$	$D^{3/4}B^{1/8}S^{1/16}S^{1/16}$

3. IMPLEMENTATION

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

[TO-DO COPY REST OVER]

4. CONCLUSION

I have presented an application of PTGGs to the topic of automated composition along the sole parameter of rhythm. I detailed one implementation of such a grammar and examined a sample of its output. The grammar shown above is a preliminary attempt, and there are some possible routes for its extension.

[need more conclusion]

4.1 Future Work

One clear choice would be to try introducing rhythmic concepts that this grammar cannot currently express. These include ties, rests, and triple (3/4) meter. The former two items suggest the introduction of more alphabet symbols: a Tied symbol, produced 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. Adding compatibility for triple meter would imply reworking Dotted as a nonterminal symbol. Another concept that seems to be missing from the current alphabet is that of an embellishment or ornamentation. Such symbols could add short note values before or after more metrically significant beats, and in so doing introduce a third 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. With certain adaptations, the proposed grammar could generate rhythms in more contemporary styles: for instance, the composition of mixed-meter passages would require a reworking of the first, Measure-based production phase, and

might be better generated additively as opposed to recursively. The grammar already supports the complex and nested tuplets which are often found in classical music of the 20th and 21st centuries, and by slackening the imposed constraint on constant total string duration, it might also be able to generate passages in free meter or non-geometric time. 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 [10]a contrapuntal phenomenon wherein shorter durations are used more frequently toward the end of a passage or phrasewithout adding other constraints to the rule set. Finally, these variations and improvements could also be useful for interfacing my 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.