Modeling and Learning Rhythm Structure

Francesco Foscarin
CNAM, Paris
francesco.foscarin@cnam.fr

Florent Jacquemard
INRIA, Paris
florent.jacquemard@inria.fr

Philippe Rigaux
CNAM, Paris
philippe.rigaux@cnam.fr

ABSTRACT

We present a model to express preferences on rhythmic structure, based on probabilistic context-free grammars, and a procedure that learns the grammars probabilities from a dataset of scores or quantized MIDI files. The model formally defines rules related to rhythmic subdivisions and durations that are in general given in an informal language. Rules preference is then specified with probability values. One targeted application is the aggregation of rules probabilities to qualify an entire rhythm, for tasks like automatic music generation and music transcription. The paper also reports an application of this approach on two datasets.

1. INTRODUCTION

In the context of music notation, rhythm is commonly modeled as a recursive subdivision of a temporal space organized in measures, beats and sub-beats. This naturally gives rise to a representation based on hierarchical structures (*aka* Rhythm Trees [1]). Moreover, this subdivision involves, at each level, choices based on the context (in particular the current metre) and on a long-established tradition of best practices. They can be expressed as rules such as the notation convention *"beam the notes in order to highlight the beat position"* [2]. Those rules (there are countless) express preferences on rhythm with different purposes (*e.g.* reduce complexity, improve readability, *etc*.), but remains at an informal level and their application is crafted in both the core of engraving software, and the expertise of their human users.

Among the applications, one can cite: engraving routines (*e.g.* MIDI import facilities) for score editors and textual score languages such as Guido or Lilypond, automated transcription, or metre detection and score analysis (with extraction of descriptors like metrical strength).

Our first goal is to adopt a formal framework to express these rules in a computational context that enables an automatic determination of rhythm structures. Our model is based on Probabilistic Context Free Grammars (PCFG), where production rules and attached weight values specify rhythmic subdivisions in a way that is both formal and close to the musical intuition. In the targeted applications,

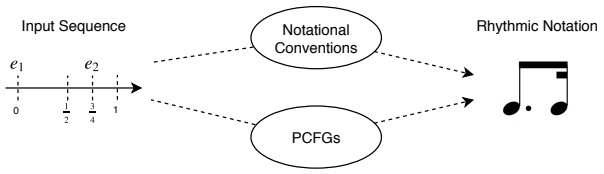

Figure 1. Modeling notational conventions with PCFGs.

the parse trees, representing the grammars' computations, also represent Rhythm Trees (hence score structure).

A PCFG acts as a model replacing informal notation rules (Figure 1). One could manually define such a model, but one can also *learn* this model, as soon as we dispose of a large enough, high quality training set (of parse trees).

An immediate thought is to base the learning step on the many corpora of existing (quantized) MIDI files or even digital scores. This gives rise however to an important issue: these datasets provide *sequences* of quantized events, but there is no direct mean to obtain the *hierarchical structures* (rhythm trees) that are necessary for learning a grammar. In other words the rhythmic representation is either missing (MIDI input) or unreliable (kern scores, or clumsy, inconsistent score publishers).

In order to overcome this limitation, we propose to produce automatically training sets from quantized input. Since there exists many possible rhythm trees that can be built from a single dataset entry, we need a decision guidelines to determine a unique candidate tree. Our decision method is based on the assumption that the *the best rhythmic representation is the one that minimizes the notation complexity.* This assumption is supported by the analysis of notation rules, and corresponds to the intuition that the purpose of a notation language is to obtain a concise, accurate and readable representation of the noted content. The main goal of the present paper is to develop an algorithm for training set production based on this assumption, and to validate it on a set of representative datasets.

The production algorithm relies on a definition of a minimization criteria, and explores the space of solutions trees that correctly represent a sequence input in order to find the minimal one with respect to this criteria. Given a dataset of sequences, we then apply the algorithm to produce the corresponding training set of minimal rhythm trees, and then carry out maximum likelihood estimation in order to obtain the PCFG (Figure 2).

Finally, we validate our method by running our training algorithm on a set of representative corpus, checking that the obtained PCFG is consistent with best notation practices. Our results confirm that the best tree decision

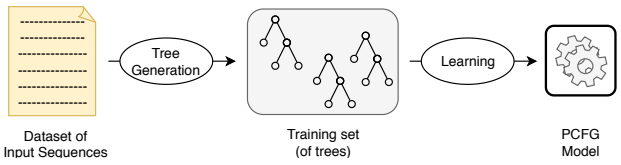

Figure 2. PCFG training from sequences of musical events.

method based on complexity minimization is a reliable computational to produce PCFG that would, otherwise, have to be manually defined.

We believe that our methodology is important for several reasons. First, PCFGs are quite useful for music transcription, and more specifically during the engraving task of producing structured notation from a quantized flow of music events. More generally, disposing of a language model is useful to measure in which extent this model is a reliable representation of the actual language used in a corpus. In concrete terms, it can be used for instance to evaluate the quality of an existing notation, or to detect outliers in a corpus (e.g., scores of MIDI files that present an unusual rhythm). Since our approach relates a grammar and the notation complexity, it also gives a quantitative measure of the rhythmic complexity of a score. In general, we consider that this constitutes a quite useful analytic tool to make sense of sequential inputs that can be structured, quantified, explored and compared.

To summarize, the paper: (1) uses PCFGs as a formalization of rhythm notation rules (Section 3), (2) learns PCFGs from datasets of music events sequences, producing training sets thanks to a complexity minimization criteria, (Section 4) and (3) validates that the resulting PCFGs trained on a dataset indeed accurately capture the best practices established in music notation (Section 5). We begin by Section 2 that briefly reviews some of the current works that use trees and grammar to work with rhythm and conclude with Section 6.

2. STATE OF THE ART

Many works in the literature rely on linear models (*e.g.*, n-grams) that apply to the sequential flow of music events [3].

Another category, more suited to represent the hierarchical structure of rhythm notation, are models based on trees and grammars. Starting from the Generative Theory of Tonal Music by Lerdahl & Jackendoff [4], those models have been successfully explored for rhythmic notation processing and evaluation [1,5,6], meter detection [7], melodic search [8] and music analysis [9–12].

In [13,14], a notation of rhythm languages defined by formal context-free grammars is proposed in order to fix the kinds of rhythmic notation to consider using declarative rules. In [15], we propose techniques based on weighted context-free grammars for automatic rhythm transcription, but the grammar is assumed given and no details are given about the procedure to construct it. In this paper we start from the same settings but we focus on the grammar creation, using results for context-free-grammars presented in [16] to obtain a model that can be trained on a dataset.

3. MODEL SPECIFICATION

Probabilistic Context-Free Grammars (PCFGs) extend CF grammars with rule probabilities. Computations of PCFGs are conveniently represented as hierarchical structures called *parse trees*. As observed in several papers, such tree structure are natural representations of common Western notation for rhythms, as they reflect structural nested decomposition of measures into beats.

3.1 Context Free Grammars for Rhythm

A PCFG is a tuple $\mathcal{G} = \langle Q, q_{\mathsf{init}}, R\rangle$ where (i) Q is a finite set of non-terminal symbols (*nt*), denoted q_0, $q_1\ldots$, (ii) $q_{\mathsf{init}} \in Q$ is a starting non-terminal, and (iii) R is a finite set of weighted production rules of one of the two following types, where w is a weight value in $[0, 1]$:

$(k{-}\mathsf{div})$ $q_0 \xrightarrow{w} q_1 \ldots q_k$ with $q_0, \ldots, q_k \in Q$ and $k > 1$,

(leaf) $q_0 \xrightarrow{w} n$ with $q_0 \in Q$ and $n \in \mathbb{N}$,

such that for all $q_0 \in Q$, $\sum_{q_0 \xrightarrow{w} \alpha \in R} w = 1$ (where α stands for $q_1, \ldots, q_k \in Q$ or $n \in \mathbb{N}$). The *nt* q_0 is called the *head* of both above rules.

A production rule $(k{-}\mathsf{div})$ describes the division of a time interval into parts of same length, *e.g.* the division of a quarter note into 2 eight notes (for $k = 2$) or into a triplet (for $k = 3$). The recursive application of $(k{-}\mathsf{div})$ rules represents nested divisions. A (leaf) rule expresses that the time interval I reached in *nt* q_0 contains n events, all aligned at the left bound of I. When $n > 1$, it means that we have $n - 1$ grace notes, of theoretical duration 0, followed by one note spanning over I. When $n = 0$, I is called a *continuation*, and its function is similar to that of a tie or a dot in music notation. Continuations are a fundamental concept in our model, since they practically allow us to split a note in multiple parts that span multiple terminal symbols.

The weight of nested divisions and event alignments is the product of the weights of all the rules involved.

Example 1. Let us consider the PCFG in Table 1. Applying the rule ρ_1 to $[0, 1[$ results in two sub-intervals $[0, \frac{1}{2}[$ and $[\frac{1}{2}, 1[$, and both of them can be processed with any rule with head $q_{\frac{1}{2}}$. Assume that we apply ρ_{11} to the first sub-interval and ρ_3 to the second one, which is then divided into $[\frac{1}{2}, \frac{3}{4}[$ and $[\frac{3}{4}, 1[$. Then, we apply respectively ρ_{16} and ρ_{17} to the latter two sub-sub-intervals of length $\frac{1}{4}$. The above rule applications result in the division of the initial interval $[0, 1[$ into a partition made of $[0, \frac{1}{2}[$, $[\frac{1}{2}, \frac{3}{4}[$ and $[\frac{3}{4}, 1[$. The first part contains a single event, at time 0, the second is a continuation (of the first event), and the last part contains a single event, at time $\frac{3}{4}$. Hence the above computation describes the rhythm represented in Figure 3.b.

Following our focus on rhythmic notation, the rules of type (leaf) only care about numbers of musical events, and contain no information about the events themselves, like pitch values, or the nature of events (note or chord). For a representation of melodies, one could replace the natural

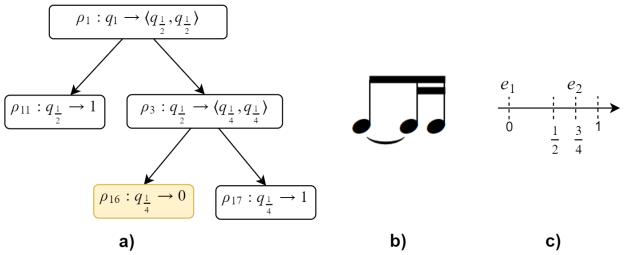

Figure 3. A parse tree (a), the respective music notation given a metric of $\frac{1}{4}$ (b) and timeline given an interval $[0, 1[$ (c). The leaf representing a *cont* is highlighted in yellow. Note that due to the *cont*, the timeline has 2 events, even if the parse tree has 3 leaves.

numbers in (leaf) rules by terminal symbols in some alphabet appropriate to the representation of musical events.

3.2 Parse Trees

We formalize the computations of a PCFG $\mathcal{G} = \langle Q, q_{\mathsf{init}}, R\rangle$ with the notion of *parse tree*, which is a tree t labeled with rules of R, such that: every inner node of t is labeled by a $(k{-}\mathsf{div})$ rule, every leaf of t is labeled by (leaf) rules, and if an inner node η is labeled by $\rho = q_0 \xrightarrow{w} q_1 \dots q_k$, then it has exactly k subtrees $t_1,\dots, t_k$ whose respective roots have heads $q_1, \dots q_k$. The subtree of t with root η is then denoted by $\rho(t_1, \dots, t_k)$. The weight $weight(t)$ of a parse tree t is the product of the weights of all the transitions labeling its nodes. It is defined recursively by $weight\big(\rho(t_1, \dots, t_k)\big) = w \times \prod i = 1^k weight(t_i)$ when ρ is $q_0 \xrightarrow{w} q_1 \dots q_k$ and $weight(\rho_0) = w$ for $\rho_0 = q_0 \xrightarrow{w} n$.

Example 2. The parse tree corresponding to the computation described in Example 1 is depicted in Figure 3.a.

3.3 Timelines and Parse Trees Serialization

We consider in the following time-points expressed in fraction of 1 measure (a rational value). A *timeline* $\ell = \langle I, \sigma\rangle$ is the representation of a sequence of events made of a left-open time interval $I = [p, p'[$ called *carrier* of ℓ and a sequence σ of time-points inside I. We assume that σ is increasing but not strictly increasing (*i.e.* it may contain repetitions). Also, the first event in σ may be distinct from the left bound p of I. In this case (and also when σ is empty), it means that ℓ starts with a continuation.

We associate to every parse tree t of a PCFG $\mathcal{G}$ and time interval I a timeline denoted $\|t\|_I$ and defined by:
$\|\rho_0\|_{[p,p'[} = \langle [p, p'[, (\underbrace{p, \dots, p}_{n})\rangle$ for $\rho_0 = q_0 \xrightarrow{w} n$, and
$\|\rho(t_1, \dots, t_k)\|_I$ is the concatenation of the timelines $\|t_1\|_{I_1}$, $\dots$, $\|t_k\|_{I_k}$, for $\rho = q_0 \xrightarrow{w} q_1 \dots q_k$, and where $I_1, \dots, I_k$ is a partition of I into k sub-intervals of equal duration. We say that a parse tree t *yields* a timeline $\ell = \langle I, \sigma\rangle$ iff $\|t\|_I = \ell$.

Therefore every parse tree t of a PCFG $\mathcal{G}$ yields an organization $\|t_k\|_I$ of events in time and also a grouping structure for these events. In other terms, t is a consistent representation of music events with respect to the notation defined by $\mathcal{G}$, and given a time signature, a music score can be constructed from it (Figure 3). We call this process *score production*. Differently from the serialization process, the continuations remain in the final result of the score production.

A parse tree can be used to represent an entire score or part of it. In this paper we represent each measure of a score with a different parse tree, *i.e.* the timeline produced by the serialization of a parse tree will represent a single measure. To summarize, we use parse trees as a model for both rhythmic structure and rhythmic notation.

4. MODEL TRAINING

We consider the problem of computing weight values in the rules of a PCFG from a dataset made of timelines. Approaches based on maximum likelihood estimator [16] permit to obtain such weights from a training set made of parse trees. Therefore, in order to apply such approaches (in Section 4.2), we need to convert datasets of timelines into training sets containing parse trees (Section 4.1).

4.1 Training Set Construction from a Score Corpus

As mentioned in the introduction we produce a training set of parse trees from a dataset of monophonic sequences extracted from a corpus of scores. This applies to a wide range of input scores, from (quantized) MIDI files to XML scores. In the latter case, one could potentially benefit from the grouping elements (beaming and tuplets) in the music notation, but this information calls for high-quality corpora where the notation complies to the best practices. Our approach holds independently from such assumptions.

From each score (or MIDI file) in the corpus, each part in the score, each voice in the part, and each measure in the voice, we extract a timeline (of carrier $[0, 1[$) from the list of event durations. We use this dataset $\mathcal{D}$ of extracted timelines as input to build parse trees. The resulting set of parse trees is the training set $\mathcal{T}$ used for learning a grammar.

Let us assume given an *acyclic* grammar $\mathcal{G} = \langle Q, q_{\mathsf{init}}, R\rangle$ whose weight are initially unknown (we call such $\mathcal{G}$ *unweighted*).

We produce for each timeline $\ell \in \mathcal{D}$ one parse tree t of $\mathcal{G}$ such that $\|t\|_{[0,1[} = \ell$, called the *representative* of ℓ in the training set. Since there exists several possible parse trees, we need a criteria to choose a unique representative.

We choose the tree t with a minimum number of leaves. This choice makes sense both from a computational and from a musical point of view. In fact, we prefer that the scores produced by $\mathcal{G}$ not to be crowded with useless notes and ties. Later in Section 5.3 we will show that the results obtained with the above criteria are coherent with some common recommendations for rhythm notation. It means that the trained PCFG will be suited to represent both rhythm and rhythm notation *wrt* such recommendations.

The following function *rep* returns for a *nt* $q \in Q$ and a timeline $\ell = \langle I, \sigma\rangle$, a parse tree t of $\mathcal{G}$, with root headed by q, yielding ℓ, and with a minimal number of leaves. In

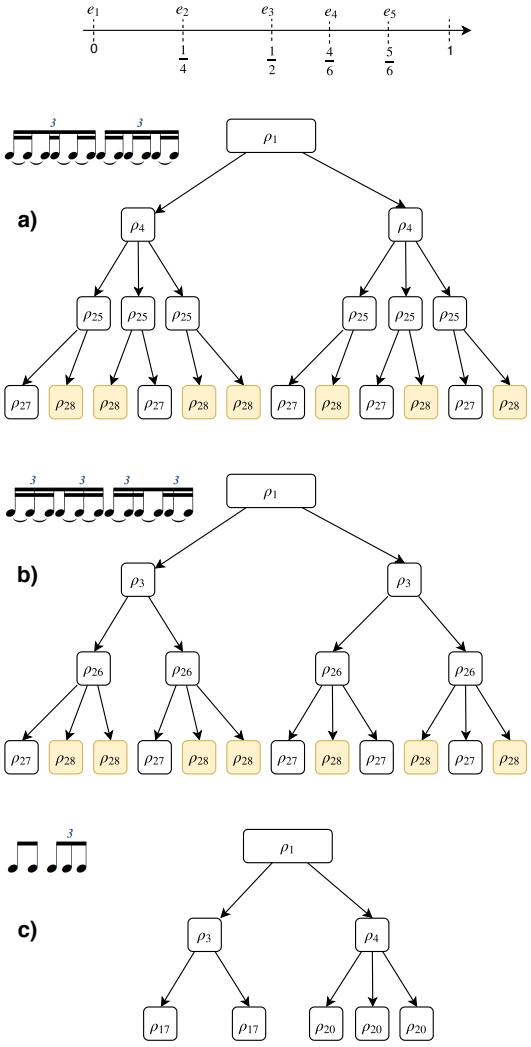

Figure 4. Three different trees for the same timeline. The terminal rules corresponding to a a continuation are highlighted in yellow.

the definition of rep, the min of a set T of trees is the tree with a minimum number of leaves. This min is undefined when T is empty or contains at least two trees with a minimum number of leaves.

If σ is empty or all points of σ coincide with the left bound of I, then

$$rep(q_0, \ell) = \rho_0 \tag{1}$$

where $\rho_0 = q_0 \to |\sigma|$, if $\rho_0 \in R$, or else $rep(q_0, \ell)$ is undefined. For the other cases of σ,

$$rep(q, \ell) = \min_{\rho = q \to (q_1, \cdots, q_k)} \Big(\rho\big(rep(q_1, \ell_1), \ldots, rep(q_k, \ell_k)\big) \Big) \tag{2}$$

where $\ell_1, \ldots, \ell_k$ is the partition of ℓ into k timelines of equal duration.

Example 3. Let us present some steps of the computation of rep for the grammar in Table 1 (forgetting the weight values), and the timeline $\ell = \langle [0, 1[, (0, \frac{3}{4}) \rangle$ of Figure 3.

$$rep(q_1, \ell) = \min \begin{cases} \rho_1\big(rep(q_{\frac{1}{2}}, \ell_{2,1}), rep(q_{\frac{1}{2}}, \ell_{2,2})\big), \\ \rho_2\big(rep(q_{\frac{1}{3}}, \ell_{3,1}), rep(q_{\frac{1}{3}}, \ell_{3,2}), rep(q_{\frac{1}{3}}, \ell_{3,3})\big) \end{cases}$$

where

$$\ell_{2,1} = \langle [0, \tfrac{1}{2}[, (0) \rangle, \ell_{2,2} = \langle [\tfrac{1}{2}, 1[, (\tfrac{3}{4}) \rangle$$
$$\ell_{3,1} = \langle [0, \tfrac{1}{3}[, (0) \rangle, \ell_{3,2} = \langle [\tfrac{1}{3}, \tfrac{2}{3}[, (\,) \rangle, \ell_{3,3} = \langle [\tfrac{2}{3}, 1[, (\tfrac{3}{4}) \rangle$$

Following (1), $rep(q_{\frac{1}{2}}, \ell_{2,1}) = \rho_{11}$, $rep(q_{\frac{1}{3}}, \ell_{3,1}) = \rho_{14}$, and $rep(q_{\frac{1}{3}}, \ell_{3,2}) = \rho_{13}$. For $rep(q_{\frac{1}{2}}, \ell_{2,2})$ and $rep(q_{\frac{1}{3}}, \ell_{3,3})$, more computation steps are needed.

The function rep can be implemented efficiently with Dynamic Programming through a tabulation procedure similar to the CYK parsing algorithm [17].

The representative of a 1-measure timeline ℓ in the dataset $\mathcal{D}$ is $rep(q_{\mathsf{init}}, \ell)$. It may be undefined, either because there is no parse tree t of $\mathcal{G}$ yielding ℓ, or, on the contrary, because there are more than one such parse trees of $\mathcal{G}$ with a minimum number of leaves. In the first case, $\mathcal{G}$ is too small to represent $\mathcal{D}$ and should be completed. The second case is discussed in Section 4.2.

This simple definition of rep is correct because the function which associate to a tree its number of leaves is monotonic, *i.e.* if t_i has more leaves than t'_i, then $\rho(t_1, \ldots, t_i, \ldots, t_k)$ has more leaves than $\rho(t_1, \ldots, t'_i, \ldots, t_k)$. It follows that we can build a best representative t for a timeline ℓ (*wrt* this criteria) from best representative for sub-timelines of ℓ (which are sub-trees of t).

4.2 Computation of Grammar's Weights

Let $\mathcal{T}$ be our training set of parse trees. We then compute the weight values for the rules $\mathcal{G}$ with a *Maximum Likelihood Estimator* [16]. For a rule $\rho = q_0 \to \alpha$, where α is either $q_1, \ldots, q_k$ or $n \in \mathbb{N}$, let $C_{\mathcal{T}}(\rho)$ be the number of occurrences of ρ in the trees of $\mathcal{T}$. The weight value for the rule ρ is then defined as $\dfrac{C_{\mathcal{T}}(\rho)}{\sum_{q_0 \to \beta \in R} C_{\mathcal{T}}(q_0 \to \beta)}$.

One can check that the grammar $\mathcal{G}'$ defined from $\mathcal{G}$ with these weight values is a PCFG, according to the definition in Section 3.1 (see [16]).

Given a timeline $\ell \in \mathcal{D}$, it may happen that its representative is undefined because there is more than one parse tree yielding ℓ with minimum number of leaves, see the example in Figure 5. In this case, it is not possible to choose a unique representative, and we will initially discard such ℓ. However, those timelines may contains useful information, and we propose the following two step procedure:

1. compute the training set $\mathcal{T}$ like in in Section 4.1, using only the timelines of $\mathcal{D}$ with a defined (unique) representative, and define weight values for $\mathcal{G}$ from $\mathcal{T}$ as above, resulting in a PCFG $\mathcal{G}'$.

2. for every timeline $\ell \in \mathcal{D}$ discarded at step 1, compute a representative t which is a parse tree of $\mathcal{G}'$ build with a modification of the function of Section 4.1 where the min *wrt* the number of leaves of t is replaced by the max of $weight(t)$. Compute new weight values with these representatives, resulting in a new PCFG $\mathcal{G}''$.

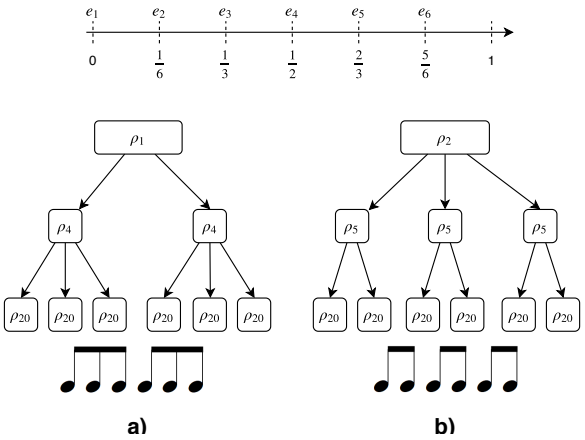

Figure 5. Two trees yielding the same timeline and corresp. notation in time signature $\frac{3}{4}$ or $\frac{6}{8}$. Both have a minimal number of leaves 6 and we cannot choose a representative between them.

5. IMPLEMENTATION

In this section we first propose two different families of unweighted acyclic grammars acting as input to the construction of Section 4, then we present the result of the grammar learning algorithm from some score corpora along with some consideration from a musical perspective.

5.1 Use-Cases of Unweighted Grammar

In theory, the learned grammar (the one given to the learning step, composed of rules without weights) must be as complete as possible. However, for practical reasons and in particular the size of a grammar that would represent *all* possible rules, we need to adopt some restrictions. In our implementation, we chose to:

1. allow $k-$div rules only with k prime number, up to a prime number K_{max}. Other $k-split$ can be obtained by sequential splits by the prime-number factors of k, *e.g.* to represent a 6-tuplet, we split by 2 and 3.

2. allow sequential rule application up to a maximum depth D_{max}, *e.g.* we stop to divide when we reach 32th notes.

3. allow only gn_{max} events in a interval, *i.e.* $gn_{max}-1$ grace-notes and one general-note.

The following are examples of practical PCFGs that respect these choices.

Example 4. A first possibility (see Table 1) is to define on rules that do not distinguish intervals of equal size in a measure, whatever their position . Thus, $[0, \frac{1}{2}[$ and $[\frac{1}{2}, 1[$ are represented by the same non terminal $q_{\frac{1}{2}}$. Each non terminal symbol represents a time interval of a specific duration, and the terminals productions specify how many events are contained in that interval (aligned to the left boundary). Given the grammar of Table 1 an informal notational rule for a $\frac{6}{8}$ metric like: "prefer to divide in 2 parts at measure level and subsequently in 3 parts" will translate in our framework in $weight(\rho_1) \geq weight(\rho_2)$ and $weight(\rho_4) \geq weight(\rho_3)$.

$$\rho_1 : q_1 \xrightarrow{0.6} \langle q_{\frac{1}{2}}, q_{\frac{1}{2}} \rangle,\ \rho_2 : q_1 \xrightarrow{0.2} \langle q_{\frac{1}{3}}, q_{\frac{1}{3}}, q_{\frac{1}{3}} \rangle$$
$$\rho_3 : q_{\frac{1}{2}} \xrightarrow{0.1} \langle q_{\frac{1}{4}}, q_{\frac{1}{4}} \rangle,\ \rho_4 : q_{\frac{1}{2}} \xrightarrow{0.7} \langle q_{\frac{1}{6}}, q_{\frac{1}{6}}, q_{\frac{1}{6}} \rangle$$
$$\rho_5 : q_{\frac{1}{3}} \xrightarrow{0.6} \langle q_{\frac{1}{6}}, q_{\frac{1}{6}} \rangle,\ \rho_6 : q_{\frac{1}{3}} \xrightarrow{0.3} \langle q_{\frac{1}{9}}, q_{\frac{1}{9}}, q_{\frac{1}{9}} \rangle$$
$$\rho_7 : q_1 \xrightarrow{0.05} 0,\ \rho_8 : q_1 \xrightarrow{0.1} 1,\ \rho_9 : q_1 \xrightarrow{0.05} 2$$
$$\rho_{10} : q_{\frac{1}{2}} \xrightarrow{0} 0,\ \rho_{11} : q_{\frac{1}{2}} \xrightarrow{0.1} 1,\ \rho_{12} : q_{\frac{1}{2}} \xrightarrow{0.1} 2$$
$$\rho_{13} : q_{\frac{1}{3}} \xrightarrow{0.05} 0,\ \rho_{14} : q_{\frac{1}{3}} \xrightarrow{0.05} 1,\ \rho_{15} : q_{\frac{1}{3}} \xrightarrow{0.0} 2$$
$$\rho_{16} : q_{\frac{1}{4}} \xrightarrow{0.1} 0,\ \rho_{17} : q_{\frac{1}{4}} \xrightarrow{0.8} 1,\ \rho_{18} : q_{\frac{1}{4}} \xrightarrow{0.1} 2$$
$$\rho_{19} : q_{\frac{1}{6}} \xrightarrow{0.3} 0,\ \rho_{20} : q_{\frac{1}{6}} \xrightarrow{0.7} 1,\ \rho_{21} : q_{\frac{1}{6}} \xrightarrow{0} 2$$
$$\rho_{22} : q_{\frac{1}{9}} \xrightarrow{0.5} 0,\ \rho_{23} : q_{\frac{1}{9}} \xrightarrow{0.3} 1,\ \rho_{24} : q_{\frac{1}{9}} \xrightarrow{0.2} 2$$

Table 1. An example of grammar with $K_{max} = 3$, $D_{max} = 2$ and $gn_{max} = 2$. Rules 1 to 9 are $k-$div rules and from 10 to 24 they are leaf rules.

The grammar above is reduced in size, but does not allow for fine-grained distinction of rules based on the position of an interval in a measure, e.g., starting on a strong beat or not. Another possibility is given below.

Example 5. Another possibility is to use a larger set of non-terminal symbols that can distinguish intervals both on the level of recursion and on the horizontal position. For instance, the first half of a measure can be treated differently from the second half, or grace notes (in leaf) rules) can be allowed for the first note of a tuplet and forbidden for the others. It allows to formally represent rules such as: "Prefer to have longer notes on stronger beats" (given that the time intervals that correspond to "stronger beats" are known), by assigning a higher probability to a leaf rule in those intervals and a higher probability of a $k-$div rule to the other intervals at the same level.

5.2 Score Corpora and Datasets

We trained the two grammars presented in the above section with the 1-measure timelines extracted from two corpora of scores: music21 corpus (`http://web.mit.edu/music21/doc/about/referenceCorpus.html`) and Enhanced Wikifonia Leadsheet Dataset (EWLD) dataset [18]. We could compute grammars for the whole dataset but there exist subsets of scores sharing some common properties that are likely to yield more consistent grammars if they are processed independently. One such property is for instance the time signature. Other possible groups could be inferred by style, tempo marking and author, depending of the level of precision that is required. We chose to divide our datasets in four subsets defined by the following time signatures: $\frac{4}{5}$, $\frac{3}{4}$, $\frac{6}{8}$, $\frac{12}{8}$. The number of scores for each group is reported in Table 2.

Within each score, we performed a simple operation of data cleaning, deleting the measure whose events durations did not sum to the correct duration given by the signature

	music21 corpus				EWLD			
	scores	measures	discarded	failed	scores	measures	discarded	failed
4/4	3261	105684	17506	139	3544	127436	3067	11554
3/4	1875	55817	9918	3228	670	30583	454	8619
6/8	2161	61048	10039	478	101	4342	44	586
12/8	9	417	129	7	129	770	10	30

Table 2. The number of scores and measures for each corpus and each time signature considered. The discarded measures are the measures whose sum does not correspond to the correct time signature; the failed measures are the one for which it was not possible to build a tree representative.

(*i.e.* pickup measures), final measures or incorrectly notated measures.

5.3 Trained PCFG from a Notational Point of View

In this section we analyze the result of the training step from a musical point of view in order to show that the criteria that we used to build unique parse trees from durations (Section 4.1) is coherent with music notation conventions.

From music general conventions [2] we know that different time signatures have different conventions about groupings in order to expose the beat (*e.g.* points were stronger accent are placed). It is interesting to notice that there is an high correlation between the probabilities learned for our grammar (the grammar of the Example 4 is sufficient for this analysis) and the divisions suggested by music notation (Table 3).

For example, music conventions state that a $\frac{3}{4}$ measure should be divided first in 3 parts (3 quarter notes); We can translate this rules in a grammar-form assigning a higher probability to the $3-$div of a measure (with respect to other $k-$div at measure level). Looking at the trees that we produced from the $\frac{3}{4}$ measures (with the algorithm in Section 4.1), we notice that this notation convention is respected, since our trees have a $3-$div at measure level in 82% of the cases. Our criteria to find the smallest tree (minimization of $yield$) allows us to build trees that are coherent with notation convention; therefore our model make sense to express rules about both rhythmic structure and rhythmic notation. From another point of view we can say that our criteria of minimizing the leaves generates results similar to that of an expert engraver who aims at making the notation as readable as possible.

6. CONCLUSIONS AND FUTURE WORKS

In this paper we presented a model of rhythm structure based on context-free-grammars and a way to learn it from a dataset of scores, addressing the problem of the generation of a training set of trees.We show that our model makes sense both from rhythm structure point of view than from a rhythmic notation point of view, comparing the frequency of the divisions in our model, with suggested divisions in music notation.

The complete implementation [1] of the grammar generation and learning is made in python and C++. The system is also partially implemented in the online digital score library NEUMA, in order to learn grammars from the corpus of scores online.

[1] Available at https://github.com/fosfrancesco/Modeling-Rhythm-Structure.git.

		music21	EWLD
4/4	1st div. by 2	99%	99%
	2nd div. by 2	98%	99%
	3rd div. by 2	93%	96%
3/4	1st div. by 3	82%	91%
	2nd div. by 2	99%	96%
6/8	1st div. by 2	70%	60%
	2nd div. by 3	90%	91%
12/8	1st div. by 2	74%	98%
	2nd div. by 2	60%	62%
	3rd div. by 3	71%	93%

Table 3. Frequency of divisions suggested in music notation in the sets of trees builded from our datasets of scores. Note that in this table the frequency have been normalized on the possible divisions, not considering terminal productions.

With big grammars (particularly with the complete grammar), we still have the problems of sparsity, *i.e.* we have lots of zeros in the final results. The best solution would be to have a bigger dataset, but alternatively the typical approach is to use a smoothing technique. However we would need to think carefully how to apply the smoothing in order to add probabilities to rare rules in a way that makes sense from a musical perspective.

The next objective is to test those grammar in a music transcription algorithm from a MIDI performance, in order to retrieve at the same time the quantized performance and the relative score.

7. REFERENCES

[1] C. Agón, K. Haddad, and G. Assayag, "Representation and rendering of rhythm structures," in *Second International Conference on WEB Delivering of Music (WEDELMUSIC)*, 2002, pp. 109–116.

[2] E. Gould, *Behind bars*. Faber Music, 2011.

[3] H. Takeda, N. Saito, T. Otsuki, M. Nakai, H. Shimodaira, and S. Sagayama, "Hidden markov model for automatic transcription of midi signals," in *IEEE Work-*

shop on Multimedia Signal Processing. IEEE, 2002, pp. 428–431.

[4] F. Lerdahl and R. S. Jackendoff, *A generative theory of tonal music*. MIT press, 1985.

[5] P. Nauert, "A theory of complexity to constrain the approximation of arbitrary sequences of timepoints," *Perspectives of New Music*, pp. 226–263, 1994.

[6] F. Jacquemard, P. Donat-Bouillud, and J. Bresson, "A structural theory of rhythm notation based on tree representations and term rewriting," in *Mathematics and Computation in Music*. Springer, 2015, pp. 3–15.

[7] A. McLeod and M. Steedman, "Meter detection and alignment of midi performance." International Conference on Music Information Retrieval (ISMIR), 2018.

[8] J. F. Bernabeu, J. Calera-Rubio, J. M. Iñesta, and D. Rizo, "Melodic identification using probabilistic tree automata," *Journal of New Music Research*, vol. 40, no. 2, pp. 93–103, 2011.

[9] M. Granroth-Wilding and M. Steedman, "Statistical parsing for harmonic analysis of jazz chord sequences," in *Proc. Intl. Computer Music Conference (ICMC)*, 2012.

[10] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.

[11] E. Nakamura, M. Hamanaka, K. Hirata, and K. Yoshii, "Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 276–280.

[12] A. Marsden, S. Tojo, and K. Hirata, "No longer'somewhat arbitrary': calculating salience in gttm-style reduction," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*. ACM, 2018, pp. 26–33.

[13] F. Jacquemard, A. Ycart, and M. Sakai, "Generating equivalent rhythmic notations based on rhythm tree languages," in *International Conference of Technologies for Music Notation and Representation (TENOR)*, 2017.

[14] A. Ycart, F. Jacquemard, J. Bresson, and S. Staworko, "A supervised approach for rhythm transcription based on tree series enumeration," in *International Computer Music Conference (ICMC)*, 2016.

[15] F. Foscarin, F. Jacquemard, P. Rigaux, and M. Sakai, "A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring," Jan. 2019, working paper or preprint. [Online]. Available: https://hal.inria.fr/hal-01988990

[16] Z. Chi and S. Geman, "Estimation of probabilistic context-free grammars," *Computational linguistics*, vol. 24, no. 2, pp. 299–305, 1998.

[17] T. Kasami, "An efficient recognition and syntax-analysis algorithm for context-free languages," Air Force Cambridge Research Lab, Bedford, MA, Tech. Rep. Scientific report AFCRL-65-758, 1965.

[18] F. Simonetta, "Enhanced wikifonia leadsheet dataset," Nov. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1476555