A Term Rewriting based Structural Theory of Rhythm Notation

Florent Jacquemard, Pierre Donat-Bouillud, Jean Bresson

A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting

Research Report of the ANR project EFFICACe *

Florent Jacquemard[1], Pierre Donat-Bouillud[1,2], and Jean Bresson[1]

[1] UMR STMS: IRCAM-CNRS-UPMC and INRIA, Paris, France.
florent.jacquemard@inria.fr, jean.bresson@ircam.fr
[2] ENS Rennes, Ker Lann Campus, France.
pierre.donat-bouillud@ens-rennes.fr

Abstract. We present a tree-based symbolic representation of rhythm notation suitable for processing with purely syntactic theoretical tools such as term rewriting systems or tree automata. Then we propose an equational theory, defined as a set of rewrite rules for transforming these representations. This theory is complete in the sense that from a given rhythm notation the rules permit to generate all notations of equivalent durations. It can be used to explore the space of Using complementary tree automata formalisms, one can restrict the search space according to notations preferences regarding e.g. metre or other user defined syntactic constraints.

Introduction

Term Rewriting Systems (TRSs) [8] are well established formalisms for tree processing (transformation and reasoning). With solid theoretical foundations, they are used in a wide range of applications, to name a few: automated reasoning, natural language processing, foundations of Web data, etc. TRSs perform in-place transformations in trees by the replacement of patterns, as defined by oriented equations called rewrite rules. They are a classical model for symbolic computation, used for rule-based modeling, simulation and verification of complex systems or software (see e.g. the languages TOM[3] and Maude[4]). Tree Automata (TAs) [7] are finite state recognizers of trees which permit to characterize specific *types* of tree-structured data (*regular* tree languages). They are often used in conjunction with TRSs, acting as *filters* in the explorations of sets of trees computed by rewriting.

It is also common to use trees to represent hierarchical structures in symbolic music (see [15] for a survey). For instance, the GTTM [13] uses trees to

* Project funded by the French National Research Agency (ANR-13-JS02-0004-01). This report is the extended version of a paper presented at the 5th International Conference Mathematics and Computation in Music (MCM 2015).

[3] http://tom.loria.fr

[4] http://maude.cs.illinois.edu

analyse inner relations in musical pieces. Trees are also a natural representation of rhythms, where durations are expressed as a hierarchy of subdivisions. Computer-aided composition (CAC) environments such as Patchwork and OpenMusic [3,6] use structures called *rhythm trees* (RTs) for representing and programming rhythms [2]. Such hierarchical, notation-oriented approach (see also [15]) is complementary to the performance-oriented formats corresponding to the MIDI notes' onsets and offsets in standard computer music systems. It also provides a more structured representation of time than music notation formats such as MusicXML [9] or Guido [12], where durations are expressed with integer values. As highly structured representations, trees enable powerful manipulation and generation processes in the rhythmic domain (see for instance [11]), and enforce some structural constraints on duration sequences.

In this paper, we propose a tree-structured representation of rhythm suitable for defining a set of rewriting rules (*i.e.* oriented equations) preserving rhythms, while allowing simplifications of notation. This representation bridges CAC rhythm structures with formal tree-processing approaches, and enables a number of new manipulations and applications in both domains. In particular, rewriting rules can be seen as an axiomatization of rhythm notation, which can be applied to reasoning on equivalent notations in computer-aided music composition or analysis.

After some preliminary definitions in Section 1, we introduce our tree representations of rhythms in Section 2. Section 3 presents the rewriting rules proposed based on this representation, and Section 4 finally states some properties of this general representation framework.

1 Preliminary Definitions

Let us assume given a countable set of variables $\mathcal{X}$, and a ranked signature Σ which is a finite set of symbols, each symbol being assigned a fixed arity. We denote as Σ_p the subset of Σ of symbols of arity p.

Trees. A Σ-labelled tree t (called *tree* for short in the rest of the paper) is either a single node, called *root* of t and denoted by $root(t)$, labeled with a variable $x \in \mathcal{X}$ or one constant symbol of $a_0 \in \Sigma_0$, or it is made of one node also denoted by $root(t)$ and labeled with a symbol $a \in \Sigma_n$ ($n > 0$), and of an ordered sequence of n *direct subtrees* $t_1, \ldots, t_n$.

In the first case, the tree t is simply denoted x or a_0. In the second case, t is denoted $a(t_1, \ldots, t_n)$, $root(t)$ is called the *parent* of respectively $root(t_1), \ldots, root(t_n)$, and the latter are called *children* of $root(t)$. Moreover, for all i, $1 < i \leq n$, $root(t_{i-1})$ is called the previous *sibling* of $root(t_i)$. For $1 \leq i \leq p$, the previous *cousin* of $root(t_i)$ is either

- $root(t_{i-1})$ if $i > 1$, or
- the last children of the previous cousin of the parent $root(t)$ if $i = 1$ and if this node exists.

Note that a tree has a previous cousin iff it has a super-tree. In other terms, the previous cousin of a node ν in a tree t is the node immediately at the left of ν in t, at the same level. A node in a tree t with no children is called a *leaf* of t. In the following, we will consider the sequence of leaves of a tree t as enumerated by a depth-first-search (*dfs*) traversal.

Example 1. Some trees are depicted in Figures 1 to 5. In the tree $2\big(\mathsf{n}, 3(\mathsf{o}, \mathsf{n}, \mathsf{n})\big)$ of Figure 2(b), the first leaf in *dfs* ordering (labeled with n) is the previous cousin of the node labeled by 3, and the second leaf in *dfs* ordering (labeled with o) is the previous sibling of the third leaf (labeled with n), which is in turn the previous sibling of the fourth leaf (also labeled with n). ◊

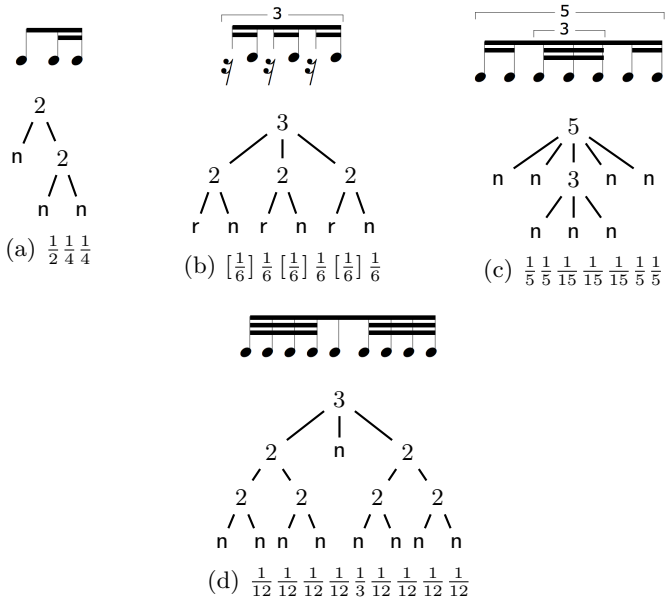

Fig. 1. Simple trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ with their corresponding rhythmic notations and values.

The set of trees built over Σ and $\mathcal{X}$ is denoted $\mathcal{T}(\Sigma, \mathcal{X})$, and the subset of trees without variables $\mathcal{T}(\Sigma)$. The definition domain of a tree $t \in \mathcal{T}(\Sigma, \mathcal{X})$, denoted by $dom(t)$, is the set of nodes of t. The *size* $|t|$ of t is the cardinality of $dom(t)$. Given $\nu \in dom(t)$: $t(\nu) \in \Sigma \cup \mathcal{X}$ is the label of ν in t, $t|_\nu$ is the subtree of t at node ν, $t[t']_\nu$ is the tree obtained from t by replacement of $t|_\nu$ by t'. We define the *depth* of a single-node tree x or a_0 as 0 and the depth of $a(t_1, \ldots, t_n)$ as 1+ the maximal depth of $t_1, \ldots, t_n$.

Pattern Matching. We call *pattern* over Σ a finite sequence of trees of $\mathcal{T}(\Sigma, \mathcal{X})$ of length $n \geq 1$, denoted as $t_1; \ldots; t_n$ (the symbol ; denotes the cousin relation). The *size* of a pattern is the sum of the sizes of its constituting trees.

A *substitution* is a mapping from variables of $\mathcal{X}$ into trees of $\mathcal{T}(\Sigma, \mathcal{X})$ with a finite domain. The application of substitutions is homomorphically extended from variables to trees and patterns: $\sigma\big(a(t_1, \ldots, t_p)\big) = a\big(\sigma(t_1), \ldots, \sigma(t_p)\big)$ and $\sigma\big(t_1; \ldots; t_n\big) = \sigma(t_1); \ldots; \sigma(t_n)$.

A tree t *matches* a pattern $t_1; \ldots; t_n$ at node ν with substitution σ if there exists a sequence of successive cousins $\nu_1, \ldots, \nu_n$ in $dom(t)$ such that $\nu_1 = \nu$ and $t|_{\nu_i} = \sigma(t_i)$ for all $1 \leq i \leq n$. When there exists such a sequence of cousins, we write $t[t'_1; \ldots; t'_n]_\nu$ for the iterated replacement $t[t'_1]_{\nu_1} \ldots [t'_n]_{\nu_n}$.

Term Rewriting. A *rewrite rule* is a pair of patterns of same length denoted $\ell_1; \ldots; \ell_n \to r_1; \ldots; r_n$ and a tree rewrite system (*TRS*) over Σ is a finite set of rewrite rules over Σ.

A tree $s \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to $t \in \mathcal{T}(\Sigma, \mathcal{X})$ with a TRS $\mathcal{R}$ over Σ, denoted by $s \xrightarrow[\mathcal{R}]{} t$ ($\mathcal{R}$ may be omitted when clear from context) if there exists a rewrite rule $\ell_1; \ldots; \ell_n \to r_1; \ldots; r_n \in \mathcal{R}$, a node $\nu \in dom(s)$ and a substitution σ over Σ such that s matches $\ell_1; \ldots; \ell_n$ at ν with σ and $t = s[\sigma(r_1); \ldots; \sigma(r_n)]_\nu$. The reflexive and transitive closure of $\xrightarrow[\mathcal{R}]{}$ is denoted by $\xrightarrow[\mathcal{R}]{*}$, and the reflexive, symmetric and transitive closure by $\xleftrightarrow[\mathcal{R}]{*}$.

This definition strictly generalizes the standard notion of term rewriting [8], which corresponds to the particular case of rewrite rules with patterns of length one (*i.e.* trees). Our notion of rewriting cousin-patterns can be captured by extensions of rewriting such as the spatial programming language MGS [5].

Tree Automata. A *tree automaton* (TA, [7]) $\mathcal{A}$ over a signature Σ is a tuple $\langle Q, \Sigma, F, \Delta \rangle$ where Q is a finite set, disjoint from Σ, of state symbols with arity 0, $F \subseteq Q$ is the subset of final states and Δ is a set of standard rewrite rules of the form: $a(q_1, \ldots, q_n) \to q$, where $a \in \Sigma_n$, and $q_1, \ldots, q_n, q \in Q$. The *language* $\mathcal{L}(\mathcal{A}, q)$ of $\mathcal{A}$ in the state $q \in Q$ is the set of trees $t \in \mathcal{T}(\Sigma)$ such that $t \xrightarrow[\Delta]{*} q$. The language of $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in F} \mathcal{L}(\mathcal{A}, q)$.

A set of trees $L \subseteq \mathcal{T}(\Sigma)$ is called *regular* if it is the language of a TA. The class of regular tree languages is effectively closed under union (construction by disjoint union of TAs), under intersection (construction by Cartesian product of TAs) and under complementation (by determination with a subset construction and inversion of final and non-final state sets). Given a TA $\mathcal{A}$ and a tree t, it is decidable in polynomial time whether $\mathcal{L}(\mathcal{A}) = \emptyset$ and whether $t \in \mathcal{L}(\mathcal{A})$.

2 Ranked Tree Representation of Rhythm Notation

We propose a tree-structured representation of rhythms based on a special ranked signature. well suited for processing with term rewriting systems

2.1 Signature

We consider a particular signature Σ_{rn} for expressing rhythm notations. It contains the following symbols of arity zero (*constant symbols*): n (representing a

note), r (rest), s (slur), d (dot), and o (for composition of durations, as explained below). Moreover, Σ_{rn} contains a subset $\mathbb{P}$ of symbols denoted as prime integers, each $p \in \mathbb{P}$ having arity p, and a copy $\bar{\mathbb{P}} = \{\bar{p} \mid p \in \mathbb{P}\}$, where $\bar{p}$ has also arity p. More precisely, $\mathbb{P}$ is assumed to contain a (small) prime integer $\max(\mathbb{P})$, assumed fixed throughout the paper, and all prime numbers smaller than $\max(\mathbb{P})$, *i.e.* $\mathbb{P} = \{2, 3, 5, \ldots, \max(\mathbb{P})\}$. Typically, $\max(\mathbb{P}) = 11$. The symbols of $\mathbb{P} \cup \bar{\mathbb{P}}$ will be used to build tuplets defined by equal subdivision of a duration. We shall also use below a set Θ_{rn} of auxiliary symbols for intermediate computation steps. It contains δ_p^i for all $p \in \mathbb{P}$ and $0 \leq i < p$, of arity p, and γ_p^i for all $p \in \mathbb{P}$ and $1 \leq i \leq p$, of arity $p - i + 1$, α_i for all $i \leq \max(\mathbb{P})$ and # of arity 1, $\perp$ and Err of arity 0.

2.2 Tree Semantics

Intuitively, a tree of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ represents a sequence of notes and rests, denoted by symbols n and r in the leaves, their duration being encoded in the structure of the tree. The symbols s and d are used to group the durations of successive leaves. The symbol o is used to group the durations of successive cousins, and possibly further subdivise the summed duration.

Formally, given a tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}}, \mathcal{X})$, we associate recursively a *duration value*, denoted $dur_t(\nu)$, to each node $\nu \in dom(t)$ as follows:

- If $\nu = root(t)$, then $dur_t(\nu)$ is a number of beats $n \geq 1$ associated to t (t can represent e.g. one or several beats or a whole bar).
- Otherwise, let ν_0 be the parent of ν in t and let p be the arity of $t(\nu_0)$, $dur_t(\nu) = \frac{dur_t(\nu_0)}{p} + cdur_t(\nu)$, where
  - $cdur_t(\nu) = dur_t(\nu')$ if ν has a previous cousin ν' such that $t(\nu') = \mathsf{o}$,
  - $cdur_t(\nu) = 0$ otherwise.

A tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}}) \setminus \{\mathsf{o}\}$ represents a sequence $val(t)$ of durations. Let $k \geq 1$ be the number of leaves of t not labeled by o and let $\nu_1 \ldots, \nu_k$ be the enumeration of these leaves in *dfs*. The *duration sequence* of t is defined as $ds(t) = \langle t(\nu_1), dur_t(\nu_1)\rangle, \ldots, \langle t(\nu_k), dur_t(\nu_k)\rangle$. Let $i_1, \ldots, i_{\ell+1}$ ($\ell \geq 0$) be an increasing sequence of indices defined as follows: $i_1 = 1$, $i_2, \ldots, i_\ell$ is the subsequence of indices of nodes in $\{\nu_2 \ldots, \nu_k\}$ labeled by n or r, $i_{\ell+1} = k + 1$.

The *rhythmic value* $val(t)$ of t is the sequence of pairs $u_1, \ldots, u_\ell$, where for each j, $1 \leq j \leq \ell$,

$$u_j = \langle t(\nu_{i_j}), \sum_{i=i_j}^{i=i_{j+1}-1} dur_t(\nu_i)\rangle$$

According to this definition, the first component of each pair u_j is either n or r, and the second component is the sum of the durations of next leaves labeled by s or d. For convenience, we shall omit the first components of pairs, and denote $val(t)$ as a sequence of durations, where the duration of rests r are written in brackets to distinguish them from durations of notes n.

Example 2. Figure 1 displays some examples of trees with the corresponding notation and rhythm value.[5] ◊

Example 3. The trees in Figure 2 contain the symbol o for the addition of durations as defined in the above semantics. Note that the duration associated to the first 3 trees is 1 beat, whereas it is 2 beats for 2(d), which represents a 2/4 bar. In the tree of Figure 2(a), the duration values of the second leaf (labeled by o) and third leaf (labeled by n) are summed to express that the second note has a duration of $\frac{1}{2}$ beat. Note that these two leaves are cousin nodes. The idea is the same in Figure 2(b) (here the second note has duration $\frac{1}{3}$). In Figure 2(c), two duration values of $\frac{1}{4}$ are also summed, like in Figure 2(a), but here, the obtained duration value of $\frac{1}{2}$ is further divided by 3. This is expressed by the 3 in the notation, which actually stands for 3 : 2 (3 *in the time of* 2). Similarly, in the bar Figure 2(d), we have 5 quavers in the time of 3. ◊

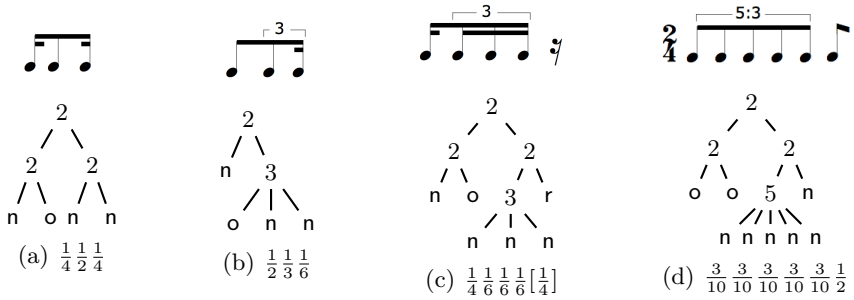

Fig. 2. Example of trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$: summation with symbol o.

Example 4. Examples of the interpretation of s and d in terms of notation are given in Figure 3. In the tree 3(a), a note of duration $\frac{1}{2}$ is dotted, extending its duration to $\frac{3}{4}$. The rhythm value of 3(b) and 3(c) is the same as for 3(a), but the notation 3(a) is more recommended (see [10]). ◊

We define as equivalent the trees representing the same actual rhythm.

Definition 1. Two trees $t_1, t_2 \in \mathcal{T}(\Sigma_{\mathsf{rn}})$ are *equivalent* iff $val(t_1) = val(t_2)$.

The tree equivalence relation is denoted $t_1 \equiv t_2$. This notion of equivalence makes it possible to characterize different notations of the same rhythm, like *e.g.* the trees (a), (b) and (c) in Figure 3.

[5] In the examples and figures of this paper, when not specified otherwise with a time signature, we will consider that the duration associated to each of the trees is 1 beat (this duration can also be found by summing the indicated fractional durations).

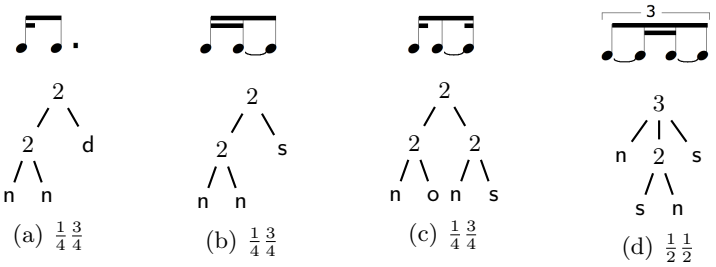

Fig. 3. Trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ with slurs and dots.

2.3 Interpretation of Trees into Common Western Notation

Our tree representation has been integrated in OpenMusic [6] with conversion functions to and from RT. We are also planning conversion to GUIDO and MusicXML. The previous examples showed how the interpretation of trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ as Common Western Notation is generally straightforward. We describe here some aspects that need particular treatments. We have already said a few words about the case of dots (symbol d), see Example 4. Let us present below another example about tuplets beaming using the symbols of $\bar{\mathbb{P}}$.

Note that the above semantics make no difference between the symbols of $\mathbb{P}$ and $\bar{\mathbb{P}}$. The symbols of $\bar{\mathbb{P}}$ are actually used to provide information on rendering beams when converting trees to musical notation.

Example 5. Figure 4 presents different ways of beaming a sextuplet (see [10]). The three trees are *equivalent*. In 4(b) and 4(c) the division is respectively bipartite and tripartite. In 4(a), the division is unclear. The symbols $\bar{2}$ and $\bar{3} \in \bar{\mathbb{P}}$ at the top of the trees 4(b) and 4(c) indicate that there must be only one beam between subtrees (the value one corresponds to the depth of $\bar{2}$ and $\bar{3}$). The default rendering, in 4(a), with label $3 \in \mathbb{P}$, is that the number of beams between subtrees is the same as the number of beams in subtrees. In Figure 5, similar variants are presented at the scale of a whole bar. Note that the value associated to each tree of Figure 5 is 4 beats, and 1 beat for each tree of Figure 4. $\Diamond$

At this point, let us make a few remarks about the above tree semantics.

1) In the tree of Figure 3(a), the dot symbol d labels a leaf node of duration $\frac{1}{2}$, which comes after another leaf node labelled by n and of duration $\frac{1}{4}$. This may seem counter-intuitive with respect to standard rhythm notation: we could expect the node of duration $\frac{1}{2}$ to be labeled by n and the node of duration $\frac{1}{4}$ to be labeled by d. However in our tree semantics, we have chosen to always represent rooted or tied notes by a n followed by some s or d, to avoid ambiguities.

2) Labels d must be used with care for ensuring a correct interpretation into notation. Section 2.4 will discuss some constraints to be satisfied for this sake.

3) The interpretation of the slur symbol s and the dot symbol d is the same regarding durations. This is also the case of $p \in \mathbb{P}$ and $\bar{p} \in \bar{\mathbb{P}}$. The symbols d and $\bar{p} \in \bar{P}$ have been introduced only to give notation-related information, and the

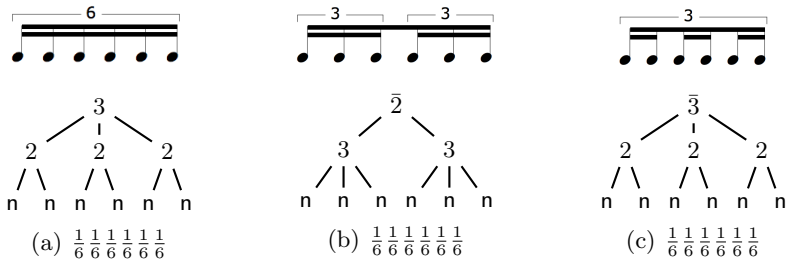

Fig. 4. Tuplet beaming with symbols of $\bar{\mathbb{P}}$ (one beat).

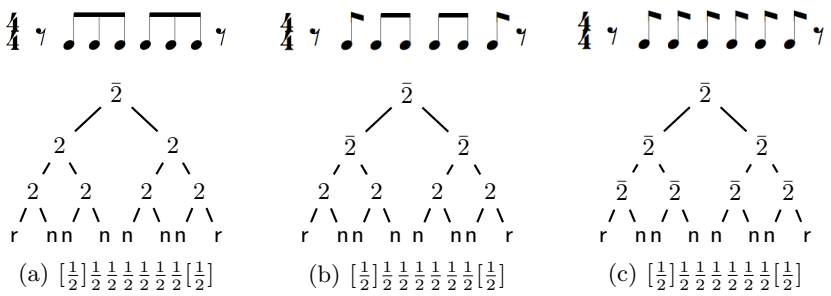

Fig. 5. Tuplet beaming with symbols of $\bar{\mathbb{P}}$ (one bar).

choice of one symbol over the other equivalent will be only dictated by notation preferences (see also Section 2.4).

4) The symbols of $\mathbb{P}$ and $\bar{\mathbb{P}}$ are somehow redundant with the tree structure, since every inner node is labeled with its degree. This technical facility however allowed us to define our trees in the algebra $\mathcal{T}(\Sigma_{\mathsf{rn}})$ over a ranked signature Σ_{rn}.

2.4 Syntactical Restrictions and Tree Automata

In general, several notations can be associated to the same rhythmic value, and the preferences regarding the notation details may depend on varied factors like the metre, usage, or personal preferences of the author. For instance, we have seen in Example 4 (Figure 3) that the dot symbol `d` produces the same rhythm value as the slur symbol `s`, but different notations. Also, separating innermost beams by using $\bar{p} \in \bar{\mathbb{P}}$ instead of $p \in \mathbb{P}$, like in Figures 4(b) and 4(c) and Figure 5, can be useful to reflect the correct subdivisions of a tuplet (following the metre) or indicate accentuations, see [10].

Some constraints regarding notation details can be expressed using tree automata (TAs) over Σ_{rn}. TAs in this case correspond to "style files" for rhythm notation.

The first and most important TA that we need in this context is the one characterizing trees with correct interpretation as notation. For instance, fol-

lowing the above remark (2), one can build a TA checking that the d symbols are well placed. This TA recognizes the set of trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ where a symbol d can only occur in a pattern of the form $2(2(x, \mathsf{n}), \mathsf{d})$ or $2(\mathsf{n}, 2(\mathsf{d}, x))$ is a regular language. We can also extend to double dots by also allowing patterns of the form $2(2(2(x, \mathsf{n}), \mathsf{d}), \mathsf{d})$ or $2(\mathsf{n}, 2(\mathsf{d}, 2(\mathsf{d}, x)))$.

Moreover, the compositional properties of TAs make it possible to combine (by union, intersection, complementation) arbitrarily different notation constraints expressed as TAs. For instance, in a binary metre, there exists a TA that allows beaming as in Figure 4(b) and forbids beaming as in Figure 4(c).

3 Rewrite Rules

We define a set of rewrite rules on trees, which do not change the rhythmic values (*i.e.* the actual rhythm), but may change its notation. These rules can therefore be used to produce equivalent notations of a same rhythm. The set of rewrite rules over Σ_{rn} defined in this section will be called $\mathcal{R}_{\mathsf{rn}}$.

3.1 Normalization Rules

The following rules reflect the semantical equivalence between dots and slurs (Example 4),

$$\mathsf{d} \to \mathsf{s} \tag{1}$$

and between symbols of $\bar{\mathbb{P}}$ and their counterpart of $\mathbb{P}$ (Example 5).

$$\bar{p}(x_1, \ldots, x_p) \to p(x_1, \ldots, x_p) \quad p \in \mathbb{P} \tag{2}$$

Addition of rests. Unlike notes, successive rests are always summed up implicitly. Following this principle, we can decide to merge subdivisions of rests, with standard rewrite rules of the form $2(\mathsf{r}, \mathsf{r}) \to \mathsf{r}$, $3(\mathsf{r}, \mathsf{r}, \mathsf{r}) \to \mathsf{r}$, *etc.*, which are generalized into

$$p(\underbrace{\mathsf{r}, \ldots, \mathsf{r}}_{p}) \to \mathsf{r} \quad p \in \mathbb{P} \tag{3}$$

The use of slurs is useless with rests, hence we have also this rule with cousin patterns of length 2

$$\mathsf{r}; \mathsf{s} \to \mathsf{r}; \mathsf{r} \tag{4}$$

Similarly, the following rule complies with the semantics of o,

$$\mathsf{o}; \mathsf{r} \to \mathsf{r}; \mathsf{r} \tag{5}$$

Normalization of s. According to the semantics of symbols of $\mathbb{P}$, we can simplify fully tied tuplets with standard rewrite rules: $2(\mathsf{s}, \mathsf{s}) \to \mathsf{s}$, $3(\mathsf{s}, \mathsf{s}, \mathsf{s}) \to \mathsf{s}$, *etc.*, generalized into

$$p(\mathsf{s}, \ldots, \mathsf{s}) \to \mathsf{s} \quad p \in \mathbb{P} \tag{6}$$

We have also $2(\mathsf{n}, \mathsf{s}) \to \mathsf{n}$, $3(\mathsf{n}, \mathsf{s}, \mathsf{s}) \to \mathsf{n}$, *etc.*, generalized into

$$p(\mathsf{n}, \mathsf{s}, \ldots, \mathsf{s}) \to \mathsf{n} \quad p \in \mathbb{P} \tag{7}$$

Normalization of o. The following rule replaces o by s when possible.

$$\mathsf{o};\mathsf{s} \to \mathsf{s};\mathsf{s} \tag{8}$$

The semantics presented in Section 2.2 make it possible to sum up the durations corresponding to cousin nodes labeled by o, and then subdivide the duration obtained by this sum. Following this principle, we can sometimes simplify a pattern beginning with a sequence of o's, according to the value of the sum and the number of subdivision. The base case is the subdivision by 1, corresponding to the atomic note n

$$\mathsf{o};\mathsf{n} \to \mathsf{n};\mathsf{s} \tag{9}$$

For a subdivision by 2, we have the following rewrite rules with variables: $\mathsf{o};2(x_1,x_2) \to x_1;x_2$, $\mathsf{o};\mathsf{o};\mathsf{o};2(x_1,x_2) \to \mathsf{o};x_1;\mathsf{o};x_2$, and so on for each multiple of 2. For a subdivision by 3, we have $\mathsf{o};\mathsf{o};3(x_1,x_2,x_3) \to x_1;x_2;x_3$ *etc.* The general form of the expected transformations is

$$\underbrace{\mathsf{o};\ldots;\mathsf{o}}_{kp-1};p(x_1,\ldots,x_p) \to \underbrace{\mathsf{o};\ldots;\mathsf{o}}_{k-1};x_1;\underbrace{\mathsf{o};\ldots;\mathsf{o}}_{k-1};x_2;\ldots;\underbrace{\mathsf{o};\ldots;\mathsf{o}}_{k-1};x_p \tag{10}$$

This transformation is simulated in several steps, using the auxiliary symbols of Θ_{rn}.
For $p=2$, we have a first left-to-right pass with

$$\begin{array}{lll}
1;2(x_1,x_2) & \to \delta_2^0(\#(x_1),\#(x_2));\bot & \\
1;\delta_2^0(x_1,x_2) & \to \delta_2^1(\alpha_1(x_1),x_2);\bot & \\
1;\delta_2^1(x_1,x_2) & \to \delta_2^0(x_1,\alpha_2(x_2));\bot & \\
\beta;\delta_2^0(x_1,x_2) & \to \beta;\gamma_2^0(x_1,x_2) & \beta \neq 1 \\
\beta;\delta_2^1(x_1,x_2) & \to \beta;\mathsf{Err} & \beta \neq 1
\end{array}$$

followed by a right-to-left pass with

$$\begin{array}{ll}
\gamma_2^0(\alpha_1(x_1),x_2);\bot & \to 1;\gamma_2^0(x_1,x_2) \\
\gamma_2^0(\#(x_1),x_2);\bot & \to x_1;\gamma_2^1(x_2) \\
\gamma_2^1(\alpha_2(x_2));\bot & \to 1;\gamma_2^1(x_2) \\
\gamma_2^1(\#(x_2)) & \to x_2
\end{array}$$

In general, the left-to-right pass involves the following rules

$$1;p(x_1,\ldots,x_p) \to \delta_p^0(\#(x_1),\ldots,\#(x_p));\bot \tag{11}$$

$$1;\delta_p^i(x_1,\ldots,x_p) \to \delta_p^j(x_1,\ldots,\alpha_{i+1}(x_{i+1}),\ldots,x_p);\bot \quad 0 \leq i < p,\ j=(i+1) \bmod p \tag{12}$$

$$\beta;\delta_p^0(x_1,\ldots,x_p) \to \beta;\gamma_p^1(x_1,\ldots,x_p) \quad \beta \neq 1 \tag{13}$$

$$\beta;\delta_p^i(x_1,\ldots,x_p) \to \beta;\mathsf{Err} \quad \beta \neq 1, i > 0 \tag{14}$$

and the second right-to-left pass is done with

$$\gamma_p^i(\alpha_i(x_i),x_{i+1},\ldots,x_p);\bot \to 1;\gamma_p^i(x_i,\ldots x_p) \quad 1 \leq i \leq p \tag{15}$$

$$\gamma_p^i(\#(x_i),x_{i+1},\ldots,x_p);\bot \to x_i;\gamma_p^{i+1}(x_{i+1},\ldots,x_p) \quad 1 \leq i < p \tag{16}$$

$$\gamma_p^p(\#(x_p)) \to x_p \tag{17}$$

Example 6. Figure 6 presents a rewrite sequence from the tree in Figure 3(c) into 3(b), and from 3(a) also into 3(b). The nodes of application of rewrite rules are marked by circles, and rewrite rules are indicated. This shows that using the above rewrite rules, we can explore the equivalent trees of Figure 3. ◊

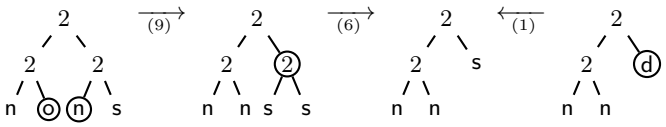

Fig. 6. Rewrite sequences starting from the trees of Figure 3(c) and 3(a). The applied rewriting rule is between parenthesis.

3.2 Subdivision Equivalence

Finally, we propose standard rewrite rules for redefining subdivisions, such as

$$\begin{aligned} 2(x_1, x_2) &\to 3(2(\mathsf{o}, \mathsf{o}), 2(x_1, \mathsf{o}), 2(\mathsf{o}, x_2)) \\ 2(x_1, x_2) &\to 5(2(\mathsf{o}, \mathsf{o}), 2(\mathsf{o}, \mathsf{o}), 2(x_1, \mathsf{o}), 2(\mathsf{o}, \mathsf{o}), 2(\mathsf{o}, x_2)) \\ 3(x_1, x_2, x_3) &\to 2(3(\mathsf{o}, x_1, \mathsf{o}), 3(x_2, \mathsf{o}, x_3)) \dots \end{aligned}$$

The general form of these rules is

$$p(x_1, \dots, x_p) \to p'\big(p(u_{1,1}, \dots, u_{1,p}), \dots, p(u_{p',1}, \dots, u_{p',p})\big) \tag{18}$$

where $p, p' \in \mathbb{P}$, $p \neq p'$, for all $1 \leq i \leq p'$, $1 \leq j \leq p$, $u_{i,j} \in \{\mathsf{o}, x_1, \dots, x_p\}$ and the sequence $u_{1,1}, \dots, u_{1,p}, \dots, u_{p',1}, \dots, u_{p',p}$ has the form $\underbrace{\mathsf{o}, \dots, \mathsf{o}, x_1}_{p'}, \dots, \underbrace{\mathsf{o}, \dots, \mathsf{o}, x_p}_{p'}$.

Example 7. Applying the above rules to the tree of Figure 3(d), we obtain the rewrite sequence depicted in Figure 7. The result is a simpler tree representing the same durations. ◊

4 Properties

We show that the rewrite rules of $\mathcal{R}_{\mathsf{rn}}$ are correct, in the sense that they preserve rhythmic values of trees, and complete, in the sense that given a tree t, it is possible to reach all trees equivalent to t using the rewriting rules of $\mathcal{R}_{\mathsf{rn}}$.

For these properties to hold, we need to consider a restriction to so called *well-formed* trees, based on the following conditions related to the symbol o. A node ν in a tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}})$ is called *dandling* if it is labeled by o and it is not the previous cousin of a node in $dom(t)$.
Let us define recursively the *simplified duration* $sdur_t$ by

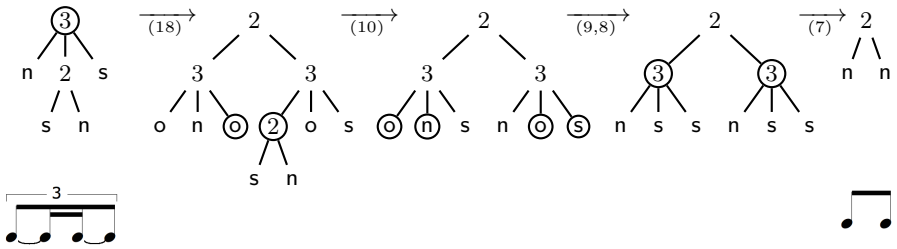

Fig. 7. Rewrite sequence starting from the tree in Figure 3(d).

$sdur_t(\nu) = dur_t(\nu)$ if $\nu = root(t)$,
$sdur_t(\nu) = \frac{sdur_t(\nu_0)}{p}$ otherwise, where ν_0 is the parent of ν and p is the arity of $t(\nu_0)$.

A tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}} \cup \Theta_{\mathsf{rn}})$ is called o-*balanced* if for all $\nu, \nu' \in dom(t)$ such that ν is the previous cousin of ν' and $t(\nu) = \mathsf{o}$, it holds that $sdur_t(\nu) = sdur_t(\nu')$. Equivalently, the multisets of labels on the two paths from ν and ν' up to the root of t are equal. Intuitively, it means that successive cousin nodes labeled by o represent the same duration.

Definition 2. A tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}})$ is *well-formed* iff it is o-*balanced* and without dandling nodes.

Example 8. All the trees of Figure 3 are well-formed. The tree 2(a) is o-balanced because the two successive leaves labeled by o and n have both a $sdur_t$ of $\frac{1}{4}$ (the multisets of labels from these nodes are both $\{2, 2\}$). The situation is the same for the tree 2(d): the three successive cousin nodes labeled by o, o and 5 all have a $sdur_t$ of $\frac{1}{4}$ (multiset $\{2, 2\}$). ◊

Note that the rhythmic value $val(t)$ is defined in Section 2.2 for any tree $t \in \mathcal{T}(\Sigma_{\mathsf{rn}})$ and not only for well-formed trees. This condition is however necessary for the correctness of rules (11-17). The set of well-formed trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ is unfortunately not regular and even not context-free [7], because it contains the set of well-balanced binary trees as a particular case.

Example 9. The tree of Figure 3(c) can be rewritten into the tree $2(2(\mathsf{n}, \mathsf{o}), \mathsf{n})$ by rule (7). The latter tree is however not well-formed because of the dandling o-node. ◊

Proposition 1. *For all well-formed trees* $t_1, t_2 \in \mathcal{T}(\Sigma_{\mathsf{rn}})$, $t_1 \equiv t_2$ *iff* $t_1 \overset{*}{\underset{\mathcal{R}_{\mathsf{rn}}}{\leftrightarrow}} t_2$.

Let us sketch the proof of Proposition 1. First, as $\mathcal{R}_{\mathsf{rn}}$ is defined over $\Sigma_{\mathsf{rn}} \cup \Theta_{\mathsf{rn}}$ and not simply Σ_{rn}, we must extend the definition of *val* and the notion of well-definition to the trees of $\mathcal{T}(\Sigma_{\mathsf{rn}} \cup \Theta_{\mathsf{rn}})$. Roughly, we need to sum the durations of contiguous o, $\perp$, s, and α_i symbols.

Moreover, we extend the definition of *val* to patterns made of o's or $\perp$'s or s's and one other tree. We show by a case analysis that for each rewrite rule $\ell_1;\ldots;\ell_n \to r_1;\ldots;r_n \in \mathcal{R}_{\mathsf{rn}}$, and for each substitution σ grounding for the rule (*i.e.* such that $\sigma(\ell_1),\ldots,\sigma(\ell_n)$, $\sigma(r_1),\ldots,\sigma(r_n)$ do not contain variables), $val\big(\sigma(\ell_1);\ldots;\sigma(\ell_n)\big) = val\big(\sigma(r_1);\ldots;\sigma(r_n)\big)$. The *if* direction of Proposition 1 then follows from a lifting of this result to the application of contexts, in order to consider rewriting at inner nodes (not only the root node).
The proof of the *only if* direction works by structural induction on t_1 and t_2.

We can use the result of Proposition 1 in order to explore rhythm notations equivalent to a given tree, for instance for simplification like in Figures 6 and 7. In our context, it is reasonable to assume a bounded depth for trees. By Kruskal lemma, it follows that the number of trees to consider is finite.

5 Conclusion

The choice of a representation determines the range of possible operations on a given musical structure, and thereby has a significant influence on compositional and analytical processes (see [11,14] for examples in the domain of rhythm structures). In this paper we proposed a formal tree-structured representation for rhythm inspired by previous theoretical models for term rewriting. Based on this representation, tree rewriting can be seen as a mean for transforming rhythms in composition or analysis processes. In a context of computer-aided composition for instance, this approach can make it possible to suggest to a user various notations of the same rhythmic value, with different complexities. Similarly, the rewrite sequence of Figure 7 can be seen as a notation *simplification* for a given rhythm. An important problem in the *confluence* of the defined rewrite relation, *i.e.* whether different rewriting from a single tree will eventually converge to a unique canonical form. For a quantitative approach, it is possible to use standard complexity measures for trees (involving depth, number of symbols *etc.*). We can therefore imagine that this framework being used as a support for rhythm quantification processes [1] in computer-aided composition environments like OpenMusic.

The tree format that we are proposing has similarities with the Patchwork/OpenMusic *Rhythmic Tree* (RT) formalism [2].[6] Still, these two formats present a number of important differences. While RTs represent durations with integers labeling nodes (the subdivision ratios), our representation only uses the tree structure (*i.e.*, the labels in $\mathbb{P}$ are not formally needed) and labels in a finite (and small) set for leaves. This specificity makes the representation more amenable to purely syntactical processing, when RT processing needs arithmetic. Trees of $\mathcal{T}(\Sigma_{\mathsf{rn}})$ and OM RTs are meant to be complementary, and some functions for converting trees back and forth between these two formats have been implemented. The definition domains of these functions are characterized by TA.

[6] A *rhythm tree* RT is defined as a pair $\langle d, S\rangle$ where $d \in \mathbb{N}$ is a duration and $S = s_0,\ldots,s_n$ is a sequence of subdivisions where for all $1 \leq i \leq n$, s_i is either a tree or a ratio. Formally, $s_i \in \mathbb{N}$ or s_i is a RT.

As mentioned in Section 2.4, it is also possible to use a TA to complement the rewriting rules and control the rhythm simplification processes by filtering out rewritten trees that do not correspond to actual notations (*e.g.* because of misplaced `d`), or/and restricting the search space to trees corresponding to acceptable or preferred notations. This approach is comparable to the use of *schemas* for XML data processing.

Note that the symbol `n` could be replaced by several symbols encoding pitches in order to represent actual melodies. Similar tree-based encodings have been used in [4] for the search of melodic similarities. Finally, other rewrite rules can be considered, including ones that do not preserve durations or rhythmic values. In this case, tree rewriting could constitute a novel creative approach to rhythm transformation in compositional applications. Another application could be the formalization of summarization of music by pruning trees like in [15].

References

1. C. Agon, G. Assayag, J. Fineberg, and C. Rueda. Kant: a Critique of Pure Quantification. In *Int. Computer Music Conf. Proc. (ICMC)*, pages 52–59, 1994.
2. C. Agon, K. Haddad, and G. Assayag. Representation and Rendering of Rhythm Structures. In *Proc. 2d Int. Conf. on WEB Delivering of Music (CW'02)*, pages 109–113, IEEE Computer Society, 2002.
3. G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue. Computer Assisted Composition at Ircam : PatchWork and OpenMusic. *Comp. Music J.*, 23(3), 1999.
4. J. F. Bernabeu, J. Calera-Rubio, J. M. Iñesta, and D. Rizo. Melodic Identification Using Probabilistic Tree Automata. *J. of New Music Research*, 40(2):93–103, 2011.
5. L. Bigo and A. Spicher. Self-assembly of musical representations in MGS. *International Journal of Unconventionnal Computing*, 2014.
6. J. Bresson, C. Agon, and G. Assayag. OpenMusic: Visual Programming Environment for Music Composition, Analysis and Research. In *Proc. of the 19th ACM international conference on MultiMedia*, pages 743–746. ACM, 2011.
7. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications.* `http://tata.gforge.inria.fr`, 2007.
8. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science*, vol. B, pages 243–320. North-Holland, 1990.
9. M. Good. Lessons from the Adoption of MusicXML as an Interchange Standard. In *XML conference*, 2006.
10. E. Gould. *Behind Bars: The Definitive Guide to Music Notation.* Faber Music, 2011.
11. K. Haddad. *Livre Premier de Motets*: The Concept of TimeBlocks in OpenMusic. In *The OM Composer's Book 2.* Editions Delatour - Ircam, 2008.
12. H. H. Hoos, K. A. Hamel, K. Renz, and K. Jürgen. Representing Score-Level Music Using the GUIDO Music-Notation Format. *Computing in Musicology*, 12, 2001.
13. F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music.* MIT Press, Cambridge, 1983.
14. M. Malt. Some Considerations on Brian Ferneyhough's Musical Language Through His Use of CAC. In *The OM Composer's Book 2.* Editions Delatour - Ircam, 2008.
15. D. Rizo. *Symbolic music comparison with tree data structures.* PhD thesis, Universidad de Alicante, November 2010.