A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring

Francesco Foscarin, Florent Jacquemard, Philippe Rigaux, Masahiko Sakai

A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring

Francesco Foscarin[1], Florent Jacquemard[2],
Philippe Rigaux[1], and Masahiko Sakai[3]

[1] CNAM Paris, France
[2] INRIA Paris, France
[3] Nagoya University, Japan

Abstract. We present a formal language-based framework for MIDI-to-score transcription, the problem of converting a sequence of symbolic musical events with arbitrary timestamps into a structured music score. The framework aims at solving in one pass the two subproblems of rhythm quantization and score production. It relies, throughout the process, on an apriori hierarchical model of scores given by generative grammars. We show that this coupled approach helps to make relevant and interrelated decisions, and we present an algorithm computing transcription solutions optimal with respect to both the fitness of the quantization to the input, and a measure of complexity of music notation.

1 Introduction

Music transcription is the act of converting a music performance into music notation (*i.e.*, a score). Several aspects of this problem are studied in the literature, according to the variety of music representation format considered. One of the most studied is the conversion of an audio recording into an unquantized MIDI file (*audio-to-MIDI*, A2M) [2], *i.e.* the extraction, from an audio signal, of a symbolic representation with explicit event descriptors such as pitch, onset and offset, expressed in a real-time unit (seconds). We focus on the complementary problem of converting unquantized MIDI into a music score (*MIDI-to-score*, M2S). M2S transcription can itself be divided into two subproblems:

(*i*) *Rhythm Quantization* (RQ) is the conversion of time values from real-time units to musical-time units (beats, or fraction of bars) [18,4,16]. RQ alone is generally achieved via the manipulation of linear data structures (e.g., sequences of messages in MIDI files).

(*ii*) *Music score production* (MSP) involves the determination of higher level information: voices, bars, grouping of events with tuplets and beams, encoding of durations with ties and dots *etc*, see [6]. A salient feature of music notation is its hierarchical nature, already advocated by the models such as Rhythm Trees [1]. An accurate MSP procedure thus requires the manipulation of hierarchical data structures.

Traditionally, subproblems (*i*) and (*ii*) are considered independently, in sequence. This allows to delegate subproblem (*ii*) to the MIDI import

module of a score editor. Such an approach to M2S transcription might be satisfying in simple cases. It strongly depends, however, on how the quantized MIDI input fits the specifics of the music notation language. To put it differently, the linear structure produced by step (i) might be hardly compatible with the hierarchical notational structures of music scores used by step (ii). Even if each step yields quite satisfying results regarding its specific goal, their combination migh therefore exhibit a discrepancy, and possibly yield poor transcription results. This is particulalry true for complex rhythms with *e.g.* deep nesting, mixed tuplets, rests, grace notes, *etc.*

In the present paper, we propose a framework for M2S where subproblems (i) and (ii) are tightly coupled: the structural information needed for score construction in (ii) is built during step (i), and takes into consideration an apriori music notation model. More precisely, (i) is solved by a *parsing algorithm*, and the parse tree defines a rhythmic structure of the output score, similar to Rhythm Tree representations [1,10].
This framework offers several distinctive advantages that contribute to improve the result quality. First, it makes it possible to jointly consider all the decisions made along the transcription process, and to model it as a *multicriteria optimization problem*, for two criterias: *fitness* between output to input, and of *rhythmic complexity*, in the sense of [17].
Second, we can leverage on expressive and powerful computational formalisms. On the one hand, we rely on *weighted context-free grammars* (WCFG), a standard formalism for modeling and ranking hierarchical constituent structure in computational linguistics. WCFG are used to describe an *a priori* music notation language. On the other hand, optimal tree representation can be obtained by efficient parsing algorithms [9] using Dynamic Programming. One of the main contribution of the paper is to show that this formal machinery can be adapted to solve M2S accurately and efficiently.

We expose the formal background of M2S in Section 2. The algorithmic part is developed in Section 3. Section 5 concludes the paper by describing experiments and further work.

2 Framework Definition and Objective

M2S takes as input a sequence of (unquantized) events and returns a music notation of this sequence. This section presents the formalisms used to represent these input and output and to model the framework.

Time Units and Tempo Curves. Timestamps can be expressed either in *real-time unit* (*rtu*), used for unquantized events, or *musical-time unit* (*mtu*) for music score events. In both cases, the temporal domain is $\mathbb{Q}_+$. In the rest of the paper, we assume a *rtu* of 1 second and a *mtu* of 1 bar. Given a time signature, every time value in mtu can be converted to a value in *beats*.
A *tempo curve* is monotonically increasing function $\theta : \mathbb{Q}_+ \to \mathbb{Q}_+$, converting rtu values into mtu values see *e.g.* [8]. Let $\bar{\tau} = \langle \tau_0, \ldots, \tau_m \rangle$ be

a sequence of rtu values with $\tau_0 = 0$ (typically the timestamps of input events). A *tempo model* $\mathcal{M}$ compatible with $\bar{\tau}$ is a set of tempo curves piecewise linear, with slope changing at the timestamps of $\bar{\tau}$. More precisely, every $\theta \in \mathcal{M}$ is defined by a sequence $\langle T_0, \ldots, T_{m-1} \rangle$ such that for all $0 \leq i \leq m-1$, the restriction of θ to $[\tau_i, \tau_{i+1}[$ is a line of slope $T_i \in \mathbb{Q}_+$ (expressed in bars per second in our case, instead of bpm). Typically $\mathcal{M}$ expresses restrictions on the changes of T_i, according to the real durations $\tau_{i+1} - \tau_i$, in order to ensure a certain smoothness of the tempo curves. We skip unecessary details about the specification of $\mathcal{M}$.

Timelines. A *time interval* is a right-open interval $I = [\tau, \tau'[\subset \mathbb{Q}_+$. I is called *unbounded* when $\tau' = +\infty$ and *bounded* otherwise. The left bound τ is called the *start* of I and denoted $start(I)$. We call *partition* of a time interval I a sequence of disjoint time intervals $I_1, \ldots, I_k$, with $k \geq 1$, such that $\bigcup_{j=1}^{k} I_k = I$. We also write $I = I_1 + \ldots + I_k$ in this case. The *k-split* of a bounded time interval I (for $k > 0$) is a partition of I of size k such that the duration of each component is $\frac{dur(I)}{k}$. In the case of a 2-split I_1, I_2 of I, we write $left(I)$ for I_1 and $right(I)$ for I_2.
We assume given a *notational alphabet* $\mathbb{E}$, which is a finite set of symbols to encode musical artifacts.

Example 1. A possible choice for $\mathbb{E}$ is the set of MIDI message symbols extended for explicit representation of rests. Pitch values are in $[0, 128]$, 128 being for rests, velocity in $[0, 127]$, and a flag distinguishes onsets/offsets. This flag, useful for polyphonic music, can be skipped in monophonic case. ◊

An *event* e is a pair $\langle \eta, \tau \rangle$ made of a symbol $\eta \in \mathbb{E}$, denoted $symb(e)$, and a *timestamp* $\tau \in \mathbb{Q}$, denoted $date(e)$. A *timeline* $\mathcal{I}$ is a pair $\langle I, \bar{e} \rangle$, where I is a time interval denoted $carrier(\mathcal{I})$ and $\bar{e} = e_1, \ldots, e_m$ is a finite sequence of events denoted $events(\mathcal{I})$, with increasing timestamps and such that $date(e_i) \in I$ for all $1 \leq i \leq m$. A timeline with timestamps in *rtu* (resp. *mtu*) is called a *real*-timeline (resp. *musical*-timeline). Operations on time intervals, like *e.g.* $+$, are extended to timelines as expected.

Example 2 (Toy running example). Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be timelines defined by: $carrier(\mathcal{I}_1) = [0, 1[$ and $events(\mathcal{I}_1) = \langle e_1, e_2, e_3 \rangle$, with respective timestamps $0.07, 0.72, 0.91$; $carrier(\mathcal{I}_2) = [1, 2[$ and $events(\mathcal{I}_2) = \langle e_4, e_5, e_6 \rangle$ with respective timestamps $1.05, 1.36, 1.71$. ◊

Semirings. Domains of weight used to rank solutions to transcription are abstractly defined as semirings, that can be instantiated into several concrete domains (*e.g.* probabilities or costs). A *semiring* $\mathcal{S} = \langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ is a structure with a domain $\mathbb{S} = dom(\mathcal{S})$, two associative binary operators $\oplus$ and $\otimes$, and neutral elements $\mathbb{0}$ and $\mathbb{1}$; $\oplus$ is commutative, $\otimes$ distributes over $\oplus$: $\forall x, y, z \in \mathbb{S}$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$, and $\mathbb{0}$ is absorbing for $\otimes$: $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$. Components of a semiring $\mathcal{S}$ may be subscripted by $\mathcal{S}$ when needed. We simply write $x \in \mathcal{S}$ to mean $x \in \mathbb{S}$.

Intuitively, in the application presented below, $\oplus$ selects an optimal value amongst two values and $\otimes$ combines two values into a single value.
A semiring $\mathcal{S}$ is *commutative* if $\otimes$ is commutative. It is *idempotent* if for all $x \in \mathcal{S}$, $x \oplus x = x$. It is *monotonic wrt* a partial ordering $\leq$ iff for all x, y, z, $x \leq y$ implies $x \oplus z \leq y \oplus z$, $x \otimes z \leq y \otimes z$ and $z \otimes x \leq z \otimes y$. Every idempotent semiring $\mathcal{S}$ induces a partial ordering $\leq_{\mathcal{S}}$ called the *natural ordering* of $\mathcal{S}$ and defined by: for all x and y, $x \leq_{\mathcal{S}} y$ iff $x \oplus y = x$. It holds then that $\mathcal{S}$ is monotonic *wrt* $\leq_{\mathcal{S}}$. $\mathcal{S}$ is called *total* if it is idempotent and $\leq_{\mathcal{S}}$ is total *i.e.* when for all x and y, either $x \oplus y = x$ or $x \oplus y = y$.
In practice, we use two kinds of total semirings: Viterbi semiring defining *probability models*, whose domain $[0, 1] \subset \mathbb{R}_+$, $\oplus$ is max, $\mathbb{0} = 0$, $\otimes$ is real product, and $\mathbb{1} = 1$, and tropical semirings, defining *cost models* whose domain $\mathbb{R}_+ \cup \{+\infty\}$, $\oplus$ is min, $\mathbb{0} = +\infty$, $\otimes$ is sum, and $\mathbb{1} = 0$.

Weighted Context-Free Grammars. A WCFG over a semiring $\mathcal{S}$ and an alphabet $\mathbb{E}$ is a tuple $\mathcal{G} = \langle Q, \mathit{init}, P, \mathit{weight}, \mathit{mus}\rangle$ where: Q is a finite set of non-terminal symbols (*nt*), $\mathit{init} \in Q$ is an initial non-terminal, P is a set of production rules in one of the following forms:
($k-$div) $q \to \langle q_1, \ldots, q_k\rangle$ with $q, q_1, \ldots, q_k \in Q$, and rank $k > 1$, or
(term) $q \to \bar{e}$ with $q \in Q$ and $\bar{e} \in \mathbb{E}^*$ ($\bar{e}$ is called *terminal* symbol).
weight assigns to each production rule in P a weight value in $\mathcal{S}$,
mus assigns to each ($k-$div) production rule in P a function associating to a musical-time interval O a partition $O_1, \ldots, O_k$ of O.

The components of a WCFG $\mathcal{G}$ may be subscripted by $\mathcal{G}$ when needed. We use the respective notations $q \xrightarrow{w} \langle q_1, \ldots, q_n\rangle$ and $q \xrightarrow{w} \bar{e}$ for ($k-$div) and (term) productions rules of weight $w \in \mathcal{S}$. The ($k-$div) rules (for $k \geq 2$) define the possible divisions of musical time intervals, *e.g.* the division of a quarter note into 2 eight notes or into a triplet. The weight associates a *rhythmic complexity* in $\mathcal{S}$ to each division. The recursive application of ($k-$div) rules represents nested divisions. Their complexity values will be composed using $\otimes$.

Example 3. The following ($2-, 3-$div) production rules, with weight values in a tropical semiring, define two possible divisions of a bounded time interval represented by the *nt* q_0, into respectively a duplet and a triplet.

$$\rho_1 : q_0 \xrightarrow{0.06} \langle q_1, q_2\rangle,\ \rho_2 : q_0 \xrightarrow{0.12} \langle q_1, q_2, q_2\rangle.$$

In those rules, q_1 represents the first event in a division, and q_2 the others. Further binary divisions of time sub-intervals are possible with:

$$\rho_3 : q_2 \xrightarrow{0.1} \langle q_3, q_3\rangle,\ \rho_4 : q_3 \xrightarrow{0.11} \langle q_4, q_4\rangle. \qquad \Diamond$$

The (term) production rules specify the musical symbols of $\mathbb{E}$ that can occur in a time interval. An empty sequence in the right-hand-side of such rule represents the *continuation* of an event started before and not yet released – notated with a tie or a dot in a score.
In practice, in order to keep $\mathcal{G}$ reasonably small, we use as set of terminal symbols a finite abstraction of $\mathbb{E}^*$, like in the following example.

Example 4. In the case of monophonic input, simultaneous events are interpreted as grace notes: a singleton sequence $\langle \eta_1 \rangle$ represents a single note, $\langle \eta_1, \eta_2 \rangle$ represents a grace note η_1 followed by a note η_2, $\langle \eta_1, \eta_2, \eta_3 \rangle$ represents two grace notes η_1, η_2 followed by a note η_3, *etc.* The set $\mathbb{F} = \{0, 1, 2, 3\}$ is a finite abstraction of $\mathbb{E}^*$ where the symbols 0, 1, 2 represent resp. a continuation, one and two symbols of $\mathbb{E}$ and 3 represents a sequence of $\mathbb{E}^*$ of length ≥ 3. In the following, we assign a weight value (in a tropical semiring) to (term) productions rules of a grammar $\mathcal{G}$, depending on the number of grace notes.

$$\begin{array}{c} \rho_5 : q_0 \xrightarrow{0.15} 0,\ \ \rho_6 : q_0 \xrightarrow{0.01} 1,\ \ \rho_7 : q_0 \xrightarrow{0.79} 2,\ \ \rho_8 : q_0 \xrightarrow{1.02} 3, \\ \rho_9 : q_1 \xrightarrow{0.02} 0,\ \rho_{10} : q_1 \xrightarrow{0.01} 1,\ \rho_{11} : q_1 \xrightarrow{0.25} 2,\ \rho_{12} : q_1 \xrightarrow{0.64} 3, \\ \rho_{13} : q_2 \xrightarrow{0.02} 1,\ \rho_{14} : q_3 \xrightarrow{0.04} 0,\ \rho_{15} : q_3 \xrightarrow{0.01} 1,\ \rho_{16} : q_4 \xrightarrow{0.01} 1 \end{array}$$

The *nt* q_0 represents a whole bar, with a single event in ρ_6 (*e.g.* a whole note in a $\frac{4}{4}$ bar), as a tied note in ρ_5 or preceded by 1 or 2 grace notes in ρ_7 and ρ_8; q_1 represents a first note in a division with a rule of Example 3 (preceded by 1 or 2 grace notes in ρ_{11} and ρ_{12}); q_2 represents the next notes in the same divisions and q_3 and q_4 further levels of divisions (grace notes are not allowed for q_2, q_3, q_4, ties are not allowed for q_4). ◊

Symbols in abstractions of $\mathbb{E}^*$ may embed more information, like *e.g.* pitch-contour for sequences of grace notes, or velocity, or on/off flag.

Parse Trees and Serialization. Given a WCTG $\mathcal{G}$ over a semiring $\mathcal{S}$, the set $\mathcal{T}(\mathcal{G})$ of *parse trees* is the smallest set of trees labelled by production rules of $\mathcal{G}$ such that:

- for all (term) rule ρ in $\mathcal{G}$, $\rho \in \mathcal{T}(\mathcal{G})$, with root ρ,
- for all ($k-$div) rule $\rho = q \xrightarrow{w} \langle q_1, \ldots, q_k \rangle$ in $\mathcal{G}$, and all $t_1, \ldots, t_k \in \mathcal{T}(\mathcal{G})$ whose respective roots have heads $q_1, \ldots, q_k$, $\rho(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{G})$ with root ρ.

In the second case, we call head of a rule its *left-hand-side nt*. We write $\mathcal{T}(\mathcal{G}, q)$ for the subset of parse trees of $\mathcal{T}(\mathcal{G})$ whose root is headed with *nt* q. The weight of a parse tree is obtained by recursively applying $\otimes$.

- $\mathit{weight}(t) := \mathit{weight}(\rho)$ when the label of t is of type (term),
- $\mathit{weight}\big(\rho(t_1, \ldots, t_k)\big) := \mathit{weight}(\rho) \otimes \big[\bigotimes_{i=1}^{k} \mathit{weight}(t_i)\big]$.

We associate to every parse tree t of $\mathcal{T}(\mathcal{G})$ and every (output) mtu interval O a musical-timeline denoted $\|t\|_O$ and defined by:

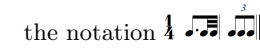
- $\|q \xrightarrow{w} a\|_O = \langle O, \langle \eta_1, \mathit{start}(O) \rangle, \ldots, \langle \eta_p, \mathit{start}(O) \rangle \rangle$, when $a = \langle \eta_1, \ldots, \eta_p \rangle$,
- $\|\rho(t_1, \ldots, t_k)\|_O = \|t_1\|_{O_1} + \ldots + \|t_k\|_{O_k}$ when $O_1, \ldots, O_k = \mathit{mus}(\rho)(O)$.

This mapping, called *serialization*, defines the timestamps for the symbols of $\mathbb{E}$ labeling the leaves of t. Moreover, t also yields a grouping structure for the resulting events. In other terms, t is a consistent representation of music events with respect to the notation defined by the grammar, and a music score can be constructed straightforwardly from it.

Example 5. Taking the rules of Examples 3, 4, assuming that $\mathit{mus}(\rho_1)$, $\mathit{mus}(\rho_2)$, $\mathit{mus}(\rho_3)$, $\mathit{mus}(\rho_4)$ return respectively 2-, 3-, 2- and 2-splits of bounded intervals, Figure 1 presents parse trees and their serialization in a 1 bar *mtu*-interval. Note that t_1 has 3 leaves, but $\|t_1\|_{[0,1[}$ contains only 2 events, because its second leaf is a continuation with 0 event. ◊

parse tree t	$\|\|t\|\|_{[0,1[}$(timestamps)	$weight(t)$	notation
$t_1 = \rho_1(\rho_{10}, \rho_3(\rho_{14}, \rho_{15}))$	$0, \frac{3}{4}$	0.22	
$t_2 = \rho_1(\rho_{10}, \rho_3(\rho_{14}, \rho_4(\rho_{16}, \rho_{16})))$	$0, \frac{3}{4}, \frac{7}{8}$	0.34	
$t_3 = \rho_2(\rho_{10}, \rho_{13}, \rho_{13})$	$0, \frac{1}{3}, \frac{2}{3}$	0.17	
$t_4 = \rho_2(\rho_{11}, \rho_{13}, \rho_{13})$	$0, 0, \frac{1}{3}, \frac{2}{3}$	0.41	

Fig. 1. Some parse trees and their linearization (Example 5).

Example 6. We extend the grammar of Example 5 with (2−div) rules $\rho_0 = q \xrightarrow{\mathbb{1}} \langle q_0, q_0 \rangle$, $\rho'_0 = q \xrightarrow{\mathbb{1}} \langle q_0, q \rangle$ for partitioning a *mtu* interval $O = [\tau, \tau'[$ into one bar $(nt\ q_0)$ and its right part $(nt\ q)$. These binary rules can be used to recursively divide a musical-time interval into several bars. The function $mus(\rho'_0)$ maps O to the partition made of $[\tau, \tau + 1[$ (first bar) and $[\tau + 1, +\infty[$ (rest of O), providing that $\tau' > \tau + 1$ or $\tau' = +\infty$. For O of duration 2 bars, the serialization $\|\rho_0(t_2, t_3)\|_O$ is a timeline with 6 events, with timestamps $0, \frac{3}{4}, \frac{7}{8}, 1, \frac{4}{3}, \frac{5}{3}$. This tree corresponds to the notation  , assuming a time signature of $\frac{1}{4}$. ◊

Input-Output Fitness Measure. We model expressive timing of human performance [8] by a composed application of a *global tempo curve* θ and *local time-shifts* for individual events. The *distance* δ measures time shifts between written and played events. It is computed in a semiring $\mathcal{S}$ based on a given $\delta_0 : \mathbb{Q}_+ \times \mathbb{Q}_+ \to \mathcal{S}$.

$$\delta(e_1, e_2) = \begin{cases} \delta_0(date(e_1), date(e_2)) & if\ symb(e_1) = symb(e_2) \\ \mathbb{0} & otherwise^4 \end{cases}$$

We extend δ to sequences of events $\bar{e} = \langle e_1, \ldots, e_m \rangle$ and $\bar{e}' = \langle e'_1, \ldots, e'_n \rangle$ by $\delta(\bar{e}, \bar{e}') = \bigotimes_{i=1}^{m} \delta(e_i, e'_i)$ if $m = n$ and $\mathbb{0}$ otherwise, and to timelines by $\delta(\mathcal{I}, \mathcal{O}) = \delta\big(events(\mathcal{I}), events(\mathcal{O})\big)$.

Example 7. For a tropical semiring, let $\delta_0(\tau_1, \tau_2) = |\tau_1 - \tau_2|$. Its extension δ is a measure of the accumulation of time-shifts of events, in rtu. ◊

We also use a measure γ of *tempo variations* defined by a tempo curve, based on a given $\gamma_0 : \mathbb{Q} \times \mathbb{Q}_+ \to \mathcal{S}$. Given a real-timeline $\mathcal{I}$, and a tempo curve θ in a model $\mathcal{M}$ compatible with $events(\mathcal{I})$, γ is defined as

$$\gamma(\theta) = \bigotimes_{i=0}^{m-1} \gamma_0(T_{i+1} - T_i, \tau_{i+1} - \tau_i)$$

where $\langle T_0, \ldots, T_{m-1} \rangle$ is the sequence of slope values defining θ (see page 2), and $\langle \tau_0, \ldots, \tau_m \rangle$ are the timestamps in $events(\mathcal{I})$.

Example 8. For a $\mathcal{S}$ tropical, we can define γ_0 as the ratio between the variation of slopes $\gamma_0(dT, d\tau) = \frac{dT}{d\tau}$ when $d\tau \neq 0$, and $\gamma_0(dT, 0) = 0$. ◊

Altogether, we define the *fitness* of a quantized musical-timeline $\mathcal{O}$ (a score) to the real-timeline $\mathcal{I}$ (a performance), *wrt* a tempo curve θ, as

$$\mathit{fit}(\mathcal{I}, \mathcal{O}, \theta) = \delta\big(\theta(\mathcal{I}), \mathcal{O}\big) \otimes \gamma(\theta).$$

In our settings, the output timeline $\mathcal{O}$ will be the serialization $\|t\|_O$ of a parse tree t of a WCFG $\mathcal{G}$ over $\mathcal{S}$ (for a given mtu time interval O).

Transcription Objective. Assuming an alphabet $\mathbb{E}$, a commutative, idempotent and total semiring $\mathcal{S}$ and a fitness measure based on δ_0 and γ_0 as above, the M2S problem is defined as follows.

INPUT:
- a real-timeline $\mathcal{I}$, non-empty (*i.e.* with $|\mathit{events}(\mathcal{I})| > 0$),
- a WCFG $\mathcal{G} = \langle \mathbb{E}^*, Q, \mathit{init}, P, \mathit{weight}, \mathit{mus} \rangle$ over $\mathcal{S}$,
- a musical-time interval $O = [0, N[$ with $N \in \mathbb{N}_+ \cup \{+\infty\}$,
- a tempo model $\mathcal{M}$ compatible with $\mathit{events}(\mathcal{I})$.

OUTPUT:
- a tempo curve $\theta \in \mathcal{M}$ defined on $\mathit{carrier}(\mathcal{I})$,
- a parse tree $t \in \mathcal{T}(\mathcal{G})$, such that $\mathit{weight}(t) \otimes \mathit{fit}(\mathcal{I}, \|t\|_O, \theta)$ is minimal *wrt* $\leq_{\mathcal{S}}$.

Therefore, the objective of M2S is to find a parse tree t representing a score that optimizes a combination of its *rhythmic complexity* (weight *wrt* $\mathcal{G}$), and its *fitness* to the input $\mathcal{I}$. The two criteria are antinomic, in the sense that improving the fitness generally increases the complexity of the tree and viceversa. Let us discuss the relevance of the above combination with $\otimes$ by reviewing two concrete domains used for for $\mathcal{S}$.

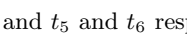
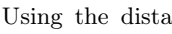
(*i*) If $\mathcal{S}$ is a Viterbi semiring, then the weight *wrt* $\mathcal{G}$ and the fitness are probability values and we want to maximize their product with $\otimes$.

(*ii*) If $\mathcal{S}$ is a tropical semiring, then the weight and the fitness can be seen as two unrelated quality criteria, and one can resettle M2S as a multicriteria optimization problem [11].

Let $\mathcal{P} = \mathcal{T}(\mathcal{G}) \times \mathcal{M}$ be the solution space for M2S, and let us consider the two objective functions c and d of $\mathcal{P}$ into the tropical semiring $\mathcal{S}$ defined, for $p = (t, \theta) \in \mathcal{P}$, by $c(p) = \mathit{weight}(t)$ and $d(p) = \mathit{fit}\big(\mathcal{I}, \|t\|_O, \theta\big)$ (for the given $\mathcal{I}$ and O). By monotonicity of $\mathcal{S}$, we can restrict our search to so-called *Pareto-optimal* points $p \in \mathcal{P}$, *i.e.* such that there is no $p' \in \mathcal{P}$ with $c(p') <_{\mathcal{S}} c(p)$ and $d(p') <_{\mathcal{S}} d(p)$.

M2S is expressed as the minimization of the combination $c(p) \otimes d(p)$, where $\otimes$ is interpreted as a sum in $\mathbb{R}_+$. This is similar to a technique of scalarization by weighted sum, selecting a point p with minimal $\alpha.c(p) + d(p)$ (in $\mathbb{R}_+$) called *scalar optimal*. Intuitively, α can be seen as a user parameter setting how much one want to favour the rhythmic complexity against the fitness. In practice, one can apply the coefficient α to $\mathit{weight}(t)$ by multiplying by α all the weight values in productions of $\mathcal{G}$. This approach is correct and complete in the following sense: every scalar optimal point $p \in \mathcal{P}$ (for some α) is Pareto-optimal and for all Pareto-optimal point $p \in \mathcal{P}$ there exists a coefficient α such that p is a scalar optimal for α (Theorem 11.17 and Corollary 11.19 of [11], chapter 11.2.1).

Example 9. Let $\mathcal{I} = \mathcal{I}_1 + \mathcal{I}_2$ (see Example 2), $\mathcal{G}$ be the WCFG from the Ex. 3,4,6, a musical-time interval $O = [0, 1[$ and a tempo model

containing a single tempo curve θ_0 mapping 1 seconds to 1 measure. Two possible solutions to M2S are the parse trees $t_5 = \rho_0(t_2, t_3)$ (Ex. 6) and $t_6 = \rho_0(t_1, t_4)$. Their serialization $\|t_5\|_O$ and $\|t_6\|_O$ have both six events with respective timestamps $(0, \frac{3}{4}, \frac{7}{8}, 1, \frac{4}{3}, \frac{5}{3})$ and $(0, \frac{3}{4}, 1, 1, \frac{4}{3}, \frac{5}{3})$; and t_5 and t_6 respectively correspond to  and  .
Using the distance defined in the Example 7 we obtain $\mathit{weight}(t_5) \otimes \mathit{fit}(\mathcal{I}, \|t_5\|_O, \theta_0) = 0.51{+}0.265 = 0.775$ and $\mathit{weight}(t_5) \otimes \mathit{fit}(\mathcal{I}, \|t_6\|_O, \theta_0) = 0.63 + 0.32 = 0.95$. That means that t_5 is preferred over t_6, and actually the algorithm of Section 3.2 will return t_5 as optimal solution. The reason is that in $\mathcal{G}$ the weight for having a grace note (rule ρ_{11}) is quite high compared to the other rules. If we lower the weight of ρ_{11} to 0.07, then $\mathit{weight}(t_6) \otimes \mathit{fit}(\mathcal{I}, \|t_6\|_O, \theta) = 0.77$ and t_6 becomes the optimal solution. This illustrates the notation preferences defined in $\mathcal{G}$, *e.g.* for favouring grace-notes or precise rhythmic notation. ◊

3 Transcription Algorithm

We now present a transcription algorithm that works in two steps: it computes first a WCFG $\mathcal{K}$ by augmenting $\mathcal{G}$ with some information from the input $\mathcal{I}$ (Sections 3.2, 3.3 describe two examples of construction of such $\mathcal{K}$, corresponding to different use cases) and then it solves M2S by computing an optimal parse tree for $\mathcal{K}$, using a Dynamic Programming algorithm presented in next Section 3.1.

3.1 Viterbi 1-best algorithm

Let $\mathcal{K}$ be a WCFG with *nt* set $\mathbb{K}$ over a total semiring $\mathcal{S}$. The following recursive function $\mathit{best}_{\mathcal{K}}$ associates to every *nt* $k \in \mathbb{K}$ a parse tree $t \in \mathcal{T}(\mathcal{K}, k)$ with a weight optimal *wrt* $\geq_{\mathcal{S}}$. By abuse, we make no distinction below between a parse tree $t \in \mathcal{T}(\mathcal{K})$ and its weight value $\mathit{weight}(t) \in \mathcal{S}$. Since $\mathcal{S}$ is total, it means that $\oplus$, applied to parse trees of $\mathcal{T}(\mathcal{K})$, selects the tree with minimal weight *wrt* $\leq_{\mathcal{S}}$.

$$\mathit{best}_{\mathcal{K}}(k) = \bigoplus_{\rho_0 = k \xrightarrow[\mathcal{K}]{w_0} a} \rho_0 \oplus \Big[\bigoplus_{\rho = k \xrightarrow[\mathcal{K}]{w} \langle k_1, \ldots, k_n \rangle} \rho\big(\mathit{best}_{\mathcal{K}}(k_1), \ldots, \mathit{best}_{\mathcal{K}}(k_n)\big) \Big]$$

If $\mathcal{K}$ is acyclic, then the above definition is well founded and the following results holds (*e.g.* by induction on k following a topological sort of $\mathbb{K}$).

Lemma 1. *For all* $k \in \mathbb{K}$, $\mathit{best}_{\mathcal{K}}(k) = \bigoplus_{t' \in \mathcal{T}(\mathcal{K}, k)} t' = \min_{\leq_{\mathcal{S}}} \{t' \in \mathcal{T}(\mathcal{K}, k)\}$.

Remember that the weight of a parse tree is the product of all the weights of the productions rules labeling its nodes. Therefore, the above formula can be understood as an alternation of sums with $\oplus$ (selection of one parse tree of optimal weight) and products with $\otimes$ (of weights of all subtrees). The function *best* can be computed by a straightforward adaptation of a Viterbi-like Dynamic Programming algorithm returning the best derivations for weighted acyclic hypergraphs ([9], Section 5.1).

With a tabulation over $\mathbb{K}$, in order to avoid recalculation of solution for subproblems, this algorithm runs in time linear in the size of $\mathcal{K}$.
In the case where $\mathcal{K}$ is not acyclic, one can use a generalization by Knuth of the Dijkstra shortest path algorithm (Algorithm 6 of [9])[5].
We apply this algorithm to a WCFG $\mathcal{K}$ built on the top of $\mathcal{G}$ from an input timeline $\mathcal{I}$. Two particular computations of $\mathcal{K}$ corresponding to different case studies are presented below.

3.2 Constant Tempo

In this first case study, we assume a constant tempo. This case study, illustrated in Example 9, corresponds to performances recorded with a metronome. The tempo model is $\mathcal{M} = \{\theta_0\}$, where a single tempo curve θ_0 represents the constant tempo value T which, for the sake of simplicity, we assume below to be the identity ($T = 1$).
In this case, the purpose of M2S transcription is essentially to correct local time-shifts of events. A parse tree t defines a partition of a given time interval O, by recursive application of the division rules labeling the nodes of t (see the definition of $\|t\|_O$). This partition can be seen as a "*grid*" containing the time positions of the bounds of the sub-intervals. M2S then consists in the *realignment* of the events of $\mathcal{I}$ to the nearest bound in the grid. The cost of this alignment is then the distance, with δ, between $\mathcal{I}$ and the score represented by t.
$\mathcal{K} = \langle \mathbb{E}^*, \mathbb{K}, \mathit{init}, P, \mathit{weight}, \mathit{mus} \rangle$ is built to represent all the time sub-intervals defined from $\mathit{carrier}(\mathcal{I})$ by recursive divisions, according to the rules of $\mathcal{G}$. For this purpose, every $nt\ k \in \mathbb{K}$ contains two components accessible with the following attributes:

- $k.nt$: a nt of $\mathcal{G}$,
- $k.car$: a real-time interval embedded in $\mathit{carrier}(\mathcal{I})$.

(div) productions rules of P are of the form $\rho = k \xrightarrow{w} \langle k_1, \ldots, k_n \rangle$, where

1. $\rho' = k.nt \xrightarrow[\mathcal{G}]{w} \langle k_1.nt, \ldots, k_n.nt \rangle$ is a rule of $\mathcal{G}$, $\mathit{mus}(\rho) = \mathit{mus}(\rho')$,
2. $k_1.car, \ldots, k_n.car$ is the application of $\mathit{mus}(\rho')$ to $k.car$,
3. $\mathit{events}(\mathcal{I}|_{k.car}) \neq \emptyset$, meaning that k is *inhabited*[6].

The last condition drastically reduces the size of $\mathcal{K}$ in practice, *wlog* since it is useless to divide empty intervals.
(term) rules of P deal with the realignement of the events of $\mathcal{I}$ to the bounds defined by a parse tree of $\mathcal{K}$, and compute the input-output distance as in Section 2. For a $nt\ k$, we know the events of $\mathcal{I}$ inside $C = k.car$. Some of these events (those in the first half of C, called *early*) will be aligned to the left bound of C. The others (those in the second half of C, called *late*) will be aligned to the right bound of C, *i.e.* actually to the left bound of the interval defined by the next leaf in a parse tree. To deal with this situation correctly, we add two components to every $nt\ k \in \mathbb{K}$, accessible with the following attributes:

- $k.post$: the late events of $\mathcal{I}$ in $k.car$ (*i.e.* those in its second half),

[5] Acyclic WCFG are however sufficient for our purpose.
[6] The restriction $\mathcal{I}|_C$ of $\mathcal{I}$ is such that by $\mathit{carrier}(\mathcal{I}|_C) = C$ and $\mathit{events}(\mathcal{I}|_C)$ is the sub-sequence of events of $\mathit{events}(\mathcal{I})$ inside C.

– $k.pre$: a buffer memorizing the *post* for the previous leaf.

And we add the following conditions for every (div) production rule:

4. $k_1.pre = k.pre, \forall 1 \leq i \leq n\ k_{i+1}.pre = k_i.post, k_n.post = k.post$.

This ensures a correct bottom-up propagation of *pre* and *post* values.
The other rules of P have the form (term) $k \xrightarrow{w} \bar{e}$, where $\bar{e} = k.pre + events\big(\mathcal{I}|_{left(k.car)}\big)$ (concatenation of the *pre* buffer and the vector of events of $\mathcal{I}$ early in $k.car$), $events\big(\mathcal{I}|_{right(k.car)}\big) = k.post$ and $w = w_0 \otimes \big[\bigotimes_{i=1}^{|\bar{e}|} \delta_0\big(start(k.car), date(e_i)\big)\big]$, with $q \xrightarrow{w_0} \bar{e}$ in $\mathcal{G}$.
The weight of the above (term) rule combines with $\otimes$ the weight w_0 of the corresponding rule in $\mathcal{G}$ (*i.e.* a complexity value) with the distance for the realignment of the points of $\bar{e}$ from their positions in $\mathcal{I}$ to new positions defined by the bounds of $k.car$. Moreover, $k_0 = init_{\mathcal{K}}$ is defined by $k_0.nt = init_{\mathcal{G}}$, $k_0.car = carrier(\mathcal{I})$, $k_0.pre = k_0.post = 0$.

Correctness and Completeness of Construction. Let us now sketch a proof that from $best_{\mathcal{K}}(k_0)$, one can build a solution for M2S conform to the definition in Section 2. First, one can observe that every parse tree $t \in \mathcal{T}(\mathcal{K})$ can be projected onto a parse tree $\pi_{\mathcal{G}}(t) \in \mathcal{T}(\mathcal{G})$. Indeed, by condition (1.) for (div) rules of $\mathcal{K}$, it is sufficient to replace, in every label of t, every *nt* k by $k.nt$ and, for (term) rules, to replace the rule's weight by the weight defined in $\mathcal{G}$. Next, for $k \in \mathbb{K}$, let use us define $\mathcal{I}|_k$ by $carrier(\mathcal{I}|_k) = carrier(\mathcal{I}|_{k.car})$ and $events(\mathcal{I}|_k) = k.pre + events\big(\mathcal{I}|_{left(k.car)}\big)$ (using the above notations).

Proposition 1. *For all $k \in \mathbb{K}$ and $t \in \mathcal{T}(\mathcal{K}, k)$, it holds that $weight(t) = weight(t') \otimes fit\big(\mathcal{I}|_k, \|t'\|_{k.car}, \theta_0\big)$, where $t' = \pi_{\mathcal{G}}(t)$.*

This can be showed by induction on t, using the above conditions (2-4) for inductions steps. It follows from Proposition 1 and the fact that $\mathcal{I}|_{k+0} = \mathcal{I}$ that $\big(\pi_{\mathcal{G}}(best_{\mathcal{K}}(k_0)), \theta_0\big)$ is a solution of M2S for $\mathcal{G}$ and $\mathcal{I}$.

3.3 Coupled Tempo Inference and Rhythm Quantization

In this second case, we sketch a construction of grammar $\mathcal{K}$ that permits a joined evaluation of a tempo curve θ and a parse tree t. Instead of realigning late events to the right bound of the current real-time interval, we move the right bound of the interval, such that the modified interval contains only early events. The new interval length induces the definition of a piecewise linear tempo curve θ like in Section 2.
In addition to the attributes $k.nt$ and $k.car$ (Section 3.2), every *nt* k of $\mathcal{K}$ has a third attribute $k.mod$ for a real-time interval modified from $k.car$. The latter is defined in (term) production rules $k \xrightarrow{w} \bar{e}$, such that events $\bar{e}$ in $k.mod$ are all early in this interval. The propagation of *mod* values for (div) rules is similar to that of *pre* and *post* (Section 3.2).
Moreover, the *nt*s of $\mathcal{K}$ also store the slope values induced by the $k.mod$, and the (div) rules also verify that the corresponding tempo curve θ (defined as in Section 2) belongs to the given model $\mathcal{M}$.

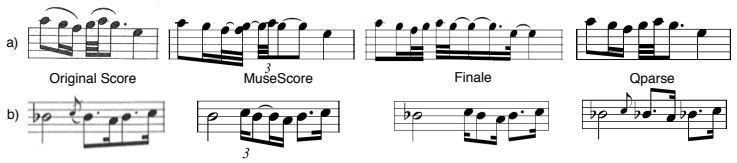


Fig. 2. Two extracts of transcription experiments; the output of our framework (qparse) is compared with the output of commercial softwares[7].

4 Experiments

The above framework for M2S transcription has been fully implemented[7] in C++. We ran manual experiments with a dataset composed of monophonic extracts from the classical repertoire, chosen from a progressive textbook for learning rhythmic reading [12]. The extracts have a length of about 15 bars on average, different time signatures (assumed known in the experiments), and contain rhythmic notations of various style and complexity. They were recorded with a MIDI keyboard by several players of different levels (non-pianist, semi-professional and professional pianists) and processed using a command line executable. The WCFGs used for experiments, ranged from generic grammars crafted manually, to specific grammars learned from the score [5].

These experiments gave promising results, especially in cases that are traditionally complex to handle (Figure 2), *e.g.* complex rhythms with mixed tuplets or alternations between short and long notes, grace notes, *etc.* The transcription time for each extract was below 100ms. Several optimisations (out of the scope of this paper) on the internal representation of grammars (with attributes) and scores (by binary trees instead of sequences) permitted important efficiency gains.

We are currently developing a general framework for automated evaluation, that stores MIDI inputs along references scores, trains WCFGs [5] from style-consistent corpus, and identifies errors in transcription output.

5 Conclusion

We have presented a modular framework for M2S transcription. It relies on a parsing algorithm for a given weighted grammar $\mathcal{G}$, and handles the problem of complex rhythms and grace notes. Grammar models contrast with the linear Markovian models used in [18,4,16]. Using hierarchical models is a trend sucessfully explored for rhythmic notation processing *e.g.* [17,1,10], meter detection [14], melodic search [3], and music analysis [7,19,15,13]. The approach is founded on the conviction that music structure complexity exceeds linear models.

An application of the same framework to polyphonic inputs is also under study, applying our framework to note-on and note-off input events to couple the voice separation problem with rhythm quantization.

[7] See `https://qparse.gitlabpages.inria.fr` for sources and complete examples.

References

1. Agon, C., Haddad, K., Assayag, G.: Representation and rendering of rhythm structures. In Proc, 2d Int. Conf. on WEB Delivering of Music (CW). pp. 109–113. IEEE Computer Society (2002).
2. Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., Klapuri, A.: Automatic music transcription: challenges and future directions. Journal of Intelligent Information Systems **41**(3), 407–434 (2013).
3. Bernabeu, J.F., Calera-Rubio, J., Iñesta, J.M., Rizo, D.: Melodic identification using probabilistic tree automata. Journal of New Music Research **40**(2), 93–103 (2011).
4. Cogliati, A., Temperley, D., Duan, Z.: Transcribing human piano performances into music notation. In: Proc. ISMIR pp. 758–764 (2016).
5. Foscarin, F., Jacquemard, F., Rigaux, P.: Modeling and Learning Rhythm Structure `https://hal.inria.fr/hal-02024437`, (2019).
6. Gould, E.: Behind Bars: The Definitive Guide to Music Notation. Faber Music (2011).
7. Granroth-Wilding, M., Steedman, M.J.: Statistical parsing for harmonic analysis of jazz chord sequences. In: Proc. ICMC (2012).
8. Honing, H.: From time to time: The representation of timing and tempo. Computer Music Journal **25**(3), 50–61 (2001).
9. Huang, L.: Advanced Dynamic Programming in Semiring and Hypergraph Frameworks. In: COLING (2008).
10. Jacquemard, F., Donat-Bouillud, P., Bresson, J.: A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting. In: Proc. MCM, Springer LNAI vol. 9110 (2015).
11. Jahn, J.: Vector Optimization. Theory, Applications, and Extensions. Springer-Verlag Berlin Heidelberg (2011).
12. Lamarque, E., Goudard, M.J.: D'un rythme à l'autre, vol. 1-4. Henry Lemoine (1997)
13. Marsden, A., Tojo, S., Hirata, K.: No Longer 'Somewhat Arbitrary': Calculating Salience in GTTM-style Reduction. In: Proc. 5th Int. Conf. on Digital Libraries for Musicology. pp. 26–33. ACM (2018).
14. McLeod, A., Steedman, M.: Meter Detection in Symbolic Music using a Lexicalized PCFG. In: proc. SMC (2017).
15. Nakamura, E., Hamanaka, M., Hirata, K., Yoshii, K.: Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In: Proc. ICASSP (2016).
16. Nakamura, E., Yoshii, K., Sagayama, S.: Rhythm transcription of polyphonic piano music based on merged-output HMM for multiple voices. IEEE/ACM TASLP **abs/1701.08343** (2017).
17. Nauert, P.: A theory of complexity to constrain the approximation of arbitrary sequences of timepoints. Perspectives of New Music **32**(2), 226–263 (1994).
18. Raphael, C.: A hybrid graphical model for rhythmic parsing. Artif. Intell. **137**(1-2), 217–238 (2002).
19. Rohrmeier, M.: Towards a generative syntax of tonal harmony. Journal of Mathematics and Music **5**(1), 35–53 (2011).