Modeling Musical Structure with Parametric Grammars

Mathieu Giraud[1] and Sławek Staworko[1,2,3]

[1] Algomus, CRIStAL (UMR CNRS 9189, Université de Lille)
[2] LINKS, Inria Lille & CRIStAL (UMR CNRS 9189, Université de Lille)
[3] Diachron project, LFCS, University of Edinburgh

Abstract. Finding high-level structure in scores is one of the main challenges in music information retrieval. Searching for a formalization enabling variety through fixed musical concepts, we use parametric grammars, an extension of context-free grammars with predicates that take parameters. Parameters are here small patterns of music that will be used with different roles in the piece. We investigate their potential use in defining and discovering the structure of a musical piece, taking example on Bach inventions. A measure of conformance of a score with a given parametric grammar based on the classical notion of edit distance is investigated. Initial analysis of computational properties of the proposed formalism is carried out.

1 Introduction

Finding high-level structure in scores is one of the fundamental research challenges in music information retrieval. Listeners are capable of discerning structure in music through the identification of common parts and their relative organization. Capturing musical structure with formal grammars is an old idea, taking roots in linguistics [1, 12, 18–20]. A grammar consists of a collection of productions, transforming *non-terminal* symbols into other symbols, and eventually producing *terminal* symbols that can be the actual notes or other elements of the musical surface. Grammars can be used as a music analysis tool, to find the right grammar modeling a piece, as well as a composition tool, to generate pieces following a grammar. Typically, for a grammar to be used as a generating tool, the productions are additionally labeled with probabilities [3].

The Shenkerian analysis [15] and the Lerdahl and Jackendoff Generative Theory of Tonal Music (GTTM) [9] share similar ideas with formal grammars. Some studies tried to put these approaches into computational models [5, 10]. Other authors use different kinds of formal languages and high-level descriptions to encode music [4,11]. Many of these formal approaches propose a derivation tree over musical surface that can be understood as parse tree of a grammar. In this view a parse tree identifies the structure of a musical piece while a grammar models the structure of a set of related pieces that share the same structural footprint. Indeed, there exists works that attempt to automatically infer a context-free grammar from a piece [17].

All such approaches may suffer from a lack of generality: The precise signification of a non-terminal is often very dependent on the piece. But it should be possible to develop more generic tools, where the concepts of *chorus*, *verse*, *theme*, *variation*, and *development* exist independently of the underlying music data. Several studies proposed techniques to infer or discover patterns used in different roles, even with some variations [2, 8]. Further research should be carried to propose formalisms able to encode notions on musical structure on potentially different musical material, and to be able to assert the compliance of these formalisms to the music. This paper proposes two steps in these directions:

- We use *parametric grammars* as an extension of context-free grammars where non-terminals have *parameters* that take values in the set of terminals. Parameters are here small patterns of music that will be used with different roles in the piece. Theses patterns are used both in measure-level or phrase-level production rules (texture arising from the local organization of the patterns) as well as in high-level organization of the piece into parts (each part involving similar or different phrase-level organizations). The formalization with parametric grammars enables us to model some musical concepts as the notion of *development* which can take several short patterns as parameters, even if the actual order of patterns is not specified. We show how to model Bach inventions with such grammars.
- We propose an algorithm to compute the optimal distance between a piece and a parametric grammar while building a derivation tree of the piece. Checking compliance between a musical piece and the model is the first step to more elaborated tasks, for instance, find the right grammar, or learning grammars from a set of examples. We show, however, that the fundamental problem of constructing the optimal alignment is intractable. To alleviate the negative impact of this result, we propose a number of practical restrictions (bounded height, width and errors) that renders computation feasible.

We prove the adequacy of our approach to model the music structure and find the optimal derivation on Bach inventions within a music analysis framework that attempts to capture only *certain* aspects of the music. We do not model everything. In fact, we do not work with single notes of the musical pieces but instead work on a level of abstraction and represent the pieces with a small set of repeated patterns. As such the generative aspect of this use of parametric grammar may be limited. The analysis perspective allows us to focus on the high-level structure. Moreover, as we try to have generic production rules in the grammar, the matching between the derivation of our grammars and the actual music is far from being exact. Nevertheless, we believe that this fuzziness reflects some aspect of the music complexity, as a fragment from any music material may have several different – and sometimes contrasting – roles.

The following Section presents the idea of the modeling on inventions by J. S. Bach. Section 3 contains a formal definition of parametric grammars and proposes a method for constructing an optimal alignment between a musical

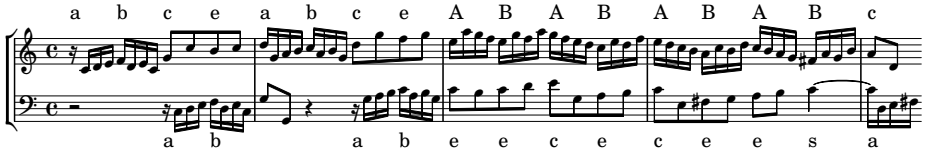

Fig. 1. Patterns used to model musical surface of Bach invention #01 in C major, taken from the first four measures of the sopran voice. All patterns have a duration of a quarter. The main patterns are a/A: four sixteenths in a upwards (a) or downwards (A) movement; b/B: four sixteenths (two successive thirds) in a upwards (b) or downwards (B) movement; c: two eights, large intervals; e: two eights, small intervals.

```
abce|abce|ABAB|ABAB|c?AB|BBzz||--ab|--ab|--AB|--AB|eece|cees| ...
--ab|--ab|eece|cees|abce|ec?z||abce|abce|ABec|ABec|ABAB|ABAB| ...

... |abAA|BbAz||ABss|abss|ABss|abss|abab|abcz|AB??|----||
... |c?AB|BBcz||--AB|ssab|ssAB|ssab|eece|ceAB|ceaz|----||
```

Fig. 2. Reduction of the whole invention using patterns of length of a quarter

piece and a parse tree that captures the structure in a musical piece. Section 4 comes back to the musical examples, presenting results of computing the parse trees that represent the music structure.

2 Modeling Bach inventions with parametric grammars

2.1 Low-level paradigmatic analysis

Most Bach inventions can be decomposed into motivic patterns, which is very convenient to test models on higher structural levels. We thus choose here to reduce the score with a paradigmatic analysis, based on our analysis and inspired by [13, 16]. Figure 1 shows the decomposition of the first four measures of both voices of Bach invention #01 using several patterns of length of a quarter note. The four main patterns (a, b, c, e) are taken from the first measure in the soprano voice. Of course, there are some arbitrary choices in this analysis – patterns could have been be longer, and many of these patterns are related. For example, pattern B is the mirror of pattern b, and pattern e can be seen as a condensed pattern B. Ultimately, most of these patterns are derived from the base patterns a and b. Figure 2 shows the decomposition of all the piece: About 80% of quarters can been seen as occurrence of either a, b, c, or e, sometimes with slight variations, including mirroring (A and B).

2.2 High-level structural analysis

We propose to roughly model the invention #01 with the following grammar G_1,

$$G_1 \quad \begin{cases} S_0() & \to P(x,y,z,w) + P(x,y,z,w) + P(z,w,x,y) \\ P(x,y,z,w) & \to T(x,y) + D(z,w) + I(w) \\ T(x,y) & \to (x/\_ + y/\_ + \_/x + \_/y) * 2 \\ & \quad \mid (\_/x + \_/y + x/\_ + y/\_) * 2 \\ D(x,y) & \to (x/\_ + y/\_) * 4 \mid (\_/x + \_/y) * 4 \\ I(x) & \to (x/\_) * 3 \mid (\_/x) * 3 \end{cases}$$

The formalism – parametric grammars – is described in the next section. The grammar ultimately generates a stream of terminals such as "$x/\_$" (pattern given by parameter x at the sopran voice, and any pattern at the alt voice). Here, we explain informally how this grammar allows to capture the structure of the invention #01.

The piece S_0 has three parts. Each part P is split in three sub-parts with an increased perceived pulse. In the *thematic* sub-part $T(x,y)$, two patterns x and y are used during two consecutive quarters, then on again two consecutive quarters but on the other voice. This pattern is repeated twice, giving a feeling of a large repeat of period one measure. For example, the first two measures (soprano: `abce|abce` / alt: `--ab|--ab`) are modeled with $T(a,b)$. In the *development* sub-part $D(x,y)$, there is a voice where the patterns x and y are repetitively used during two measures (such as `ABAB|ABAB` / `eece|cees` with $D(A,B)$). Here, the individual *halves* of each measure become more pronounced. Finally, in the *intensification* sub-part $I(x)$, there is a voice where a pattern x is played three times *at every quarter note* (such as `BBB` / `eec` with $I(B)$). This intensification is concluded by a cadential element before the start of the following part.

What are the advantages of using parametric grammar over using the standard context-free grammars? The P_1 part could be roughly defined by explicit rules of the following standard context-free grammar, without parameters:

$$G_{non\text{-}parametric} \quad \begin{cases} S_0 & \to P_1 + P_2 + P_3 \\ P_1 & \to T_1 + D_1 + I_1 \\ T_1 & \to (a/\_ + b/\_ + \_/a + \_/b) * 2 \\ D_1 & \to (a/\_ + b/\_) * 4 \\ I_1 & \to (B/\_) * 3 \\ & \dots \end{cases}$$

Similar modeling can be done for the others parts. The complete non-parametric grammar resulting from this modeling is, however, less concise, does not allow to capture the structural similarities in elements of the same function (part, theme, development, intensification), and fails to identify the connections among structural elements of the piece that are established by using the same of pieces of musical material.

In the parametric grammar, the rule deriving TDI sub-parts from $P(x,y,z,w)$ links the different sub-parts of a part. The fact that the same or similar musical

material is used throughout the part contributes the unity of this part. For instance, the pattern x for the D sub-part is reused in the I sub-part. Similarly, at the top level S_0, the different parts use the same material, but the last part, $P(z, w, x, y)$, uses the music material in a different order.

The parametric grammar G_1 can be made more flexible and closer to actual pieces by relaxing some production rules as in the following grammar G_2:

$$G_2 \quad \begin{cases} S_0() & \to P(x,y,z,w) + P(x,y,z,w) + P(z,w,x,y) & \\ P(x,y,\{z,w\}) & \to T(x,y) + D(z,w) + I(w) & \leq 2 \\ P(x,y,\{z,w\}) & \to T(x,y) + I(w) & \leq 2 \\ P(x,y,\{z,w\}) & \to T(x,y) + D(z,w) & \leq 2 \\ T(x,y) & \to (x/_- + y/_- + _-/x + _-/y) * [1;2] & \\ & \quad \mid (_-/x + _-/y + x/_- + y/_-) * [1;2] & \\ D(x,y) & \to (x/_- + y/_-) * [3;4] \mid (_-/x + _-/y) * [3;4] & \\ I(x) & \to (x/_-) * [3;4] \mid (_-/x) * [3;4] & \end{cases}$$

Now a P part can be composed from only TI or TD sub-parts instead of all three TDI sub-parts. Moreover, the number of repeats in the individual T, D and I sub-parts is variable. To limit the combinatorial explosion, a limit has been set to 2 for each P part. This bounds the "alignment distance" to the actual musical content, also limiting the number of candidate P "fragments" inside the score. Distance and fragments are formally defined in the next section.

Note also that the w pattern playing a specific role in the I rule may be any of the two patterns of the D rule. This choice is here modeled by the set $\{z, w\}$ in the rules for P.

	length	complete piece	part P_1	part P_2	part P_3	part P_4
#01 C major	88	$S_0 \to P_1P_2P_3$	TDI	$TTDI$	TD	
#03 D major	65	$S_0 \to P_1P_2P_3WP_4C$	TI	TI	TI	TI
#04 D minor	54	$S_0 \to P_1P_2P_3$	TDD	$TDDD$	T	
#13 A minor	104	$S_0 \to P_1P_2RP_3P_4$	TDI	TDI	TR	TI

Table 1. Reference analysis derivation for some Bach inventions. These inventions were manually modeled as successive parts (P) composed of thematic (T), and possibly development (D) and intensification (I) sub-parts. Some structures also include special transition (W, R) and coda (C) parts that will not be discussed here.

Table 1 details the structure of four Bach inventions that can be seen as productions of this grammar. As flexibility is inherent to the proposed grammar model, it can define a large number of different musical pieces and many, if not most, are unlikely to satisfy any reasonable aesthetic requirements of *good music*. Using parametric grammars for generative purposes would therefore require adding constraints on the flow of the piece and then generating a piece that satisfies them. This is, however, beyond the scope of this paper whose main focus is

to provide an analytic framework capable of exploring certain high-level aspects of the musical structure.

3 Parametric grammars

In this section, we formally define parametric grammars and we present how to align scores to parse trees of parametric grammars. A parametric grammar is essentially an extension of context-free grammar whose non-terminals take *parameters*. They can be viewed as a specialized attribute grammars [7]. They have the same expressiveness as context-free grammars but can be significantly more concise [6].

3.1 Definitions

Let Σ be a finite set of symbols and $k > 0$ the number of output voices. A (k-*voice*) *output atom* is an vector of k symbols i.e., an element of Σ^k, and we use $\bar{a}, \bar{b}, \ldots$ to range over output atoms. For an output atom $\bar{a} \in \Sigma^k$ by a_i we denote the symbol at the i-th voice of a i.e., $\bar{a} = (a_1, \ldots, a_k)$. A ($k$-*voice*) *string* is a sequence of atoms i.e., an element of $(\Sigma^k)^*$, and we use $w, v, \ldots$ to range over strings. By $|w|$ we denote the length of the string w and by w_i we denote the i-th atom of w for $i \in \{1, \ldots, |w|\}$ i.e., $w = (w_1, \ldots, w_{|w|})$.

A *grammar signature* is a tuple $\mathcal{S} = (\Sigma, k, X, V, \mathit{arity})$, where Σ is a finite set of symbols, $k > 0$ is the number of output voices, X is a finite set of parameters, and V is a finite set of non-terminals together with the function $\mathit{arity} : V \to \mathbb{N}$ that assigns to every transition symbol the number of its parameters. We assume a fixed grammar signature $\mathcal{S}$ and define a number of concepts over $\mathcal{S}$. An *output term* is a vector of k elements from $\Sigma \cup X$. A *intermediate term* is $N(\tau_1, \ldots, \tau_n)$, where $N \in V$ is a non-terminal of arity $n = \mathit{arity}(N)$ and $\tau_i \in \Sigma \cup X$ for $i \in \{1, \ldots, n\}$. A *term* is either an output term or a intermediate term. A term is *ground* if it does not use any parameter. Note that a ground output term is an output atom i.e., an element of Σ^k, and similarly, a sequence of ground output atoms is a string i.e., an element of $(\Sigma^k)^*$. A *substitution* is a function θ that maps parameters in X to symbols in Σ (this function may be partial). The result of applying a substitution θ to t, in symbols $\theta(t)$, is obtained by replacing every parameter x by the symbol $\theta(x)$ that θ assigns to x. Applying substitution is extended to sequences of terms in the canonical fashion: we apply the substitution to every element of the sequence. For example, on the grammar G_2, the output atoms are $\{a, b, c, e, A, B, ...\}$, the intermediate terms are $P(...)$, $T(...)$, $D(...)$ and $I(...)$, and the parameters are $\{x, y, z, w\}$. The substitution mapping $T(x, y)$ to $T(a, b)$ is $\{\theta(x) = a, \theta(y) = b\}$.

A *parametric grammar* is a tuple $G = (\mathcal{S}, S_0, P)$, where $\mathcal{S} = (\Sigma, k, X, V, \mathit{arity})$ is its signature, $S_0 \in V$ is a distinguished *starting non-terminal*, and P is a set of productions of the form $t \to s$, where t is a term and s a sequence of terms. A *derivation tree* of G is a tree T such that:

1. the leaves of T are labeled by ground output terms;
2. the non-leaf nodes are labeled by ground intermediate terms;
3. for every non-leaf node there exists a production $t \to s$ and a valuation θ such that, the node is labeled by $\theta(t)$ and the consecutive labels of its children give the sequence $\theta(s)$.

The *foliage* of a derivation tree T, denoted $yield(T)$, is the sequence obtained by taking the leaf labels in the standard left-to-right traversal of the tree. Note that $yield(T) \in (\Sigma^k)^*$ since the leaves of a derivation tree can be labeled by ground output terms only. A *parse tree* of $w \in (\Sigma^k)^*$ (w.r.t. G) is any derivation tree T whose root node is labeled by an intermediate term $S_0(a_1, \ldots, a_k)$ for some $a_i \in \Sigma$ and $yield(T) = w$. The *language* of G, denoted $L(G)$, is the set of all strings $w \in \Sigma^*$ that have a parse tree (w.r.t. G).

The grammar G_2 defined in the previous section further uses elements of syntactic sugar, allowing to represent a parametric grammar in a compact way:

1. disjunction e.g., $t \to s_1 \mid s_2$ equivalent to two rules $t \to s_1$ and $t \to s_2$,
2. numerical repetition e.g., $t \to s * [1;2]$ equivalent to $t \to s|s + s$ ($+$ is concatenation operator, omitted in the formal definition). Many repetitions in music have between 2 and 4 occurrences.
3. grouped set of parameters on the left-hand side e.g., $N(x_1, \{x_2, x_3\}) \to s$ equivalent to $N(x_1, x_2, x_3) \to s$ and $N(x_1, x_3, x_2) \to s$, as well on the right-hand side e.g., $t \to N(x_1, \{x_2, x_3\})$ equivalent to $t \to N(x_1, x_2, x_3)$ and $t \to N(x_1, x_3, x_2)$. Indeed, the same basic music material is often used in differents parts *with different roles*. For example, in the grammar G_2, the secondary patterns $\{z, w\}$ play different roles in the three P parts.

3.2 Aligning scores to parse trees

While a parametric grammar allows to define strings, hence scores, that exhibit a very specific structure defined by the grammar, real-life musical pieces rarely adhere to this structure. Consequently, we propose a method of aligning a given string w, that represents the musical surface, to a parse tree T of another string $v \in L(G)$, that exhibits the structure defined by G. In particular, we do not assume that w is recognized by G, which may be too strict, but instead we introduce a measure of distance between w and T that we aim to minimize. This measure captures two types of operations performed on the parse tree T and the string v:

1. Basic *string editing operations*, which include inserting and deleting an atom in v as well as renaming a symbol of a single voice. For example, the cost of renaming one symbol can be equal to 1 and the cost of inserting and deleting an atom can be equal to k, the number of voices. The cost can also be linked to the actual musical content of the pattern.

2. *Move operations* that introduce gaps or overlaps between the outputs (foliage) of two sibling nodes in the parse tree T. While gaps (overlaps) in T can be captured with insertion (deletion resp.) operations in v, their cost can be different. Setting this cost to something smaller than the length of the gap will favour the move of these output blocks.

We now fix a k-voice parametric grammar G and a string $w \in (\Sigma^k)^*$ that needs not belong to $L(G)$. Let $w = (w_1, \ldots, w_n)$ and define the set of *positions* in w as $Pos(w) = \{1, \ldots, n+1\}$ with $n+1$ denoting a virtual end-of-string position. A *fragment* of w is pair $F = (s, e) \in Pos(w)^2$ of positions of w such that $s \leq e$ and $start(F) = s$ is called its *start* of F and $end(F) = e$ its *end*. The fragment F represents a substring $(w_{start(F)}, w_{start(F)+1}, \ldots, w_{end(F)-1})$ and we point out that the end position of F is not included in the substring w_F but intuitively it is the position of w that immediately follows the last position of w_F. A fragment is *empty* if its start and end are the same. We define a number of relations on pairs of fragments F and F' of w: 1) F' is *included* in F if $start(F) \leq start(F')$ and $end(F') \leq end(F)$; 2) F and F' *overlap* if $start(F) \leq start(F') < end(F)$ or $start(F') \leq start(F) < end(F')$; 3) F' *follows* F if $start(F) \leq start(F')$. Note that if F' follows F, the fragments may overlap.

Now, let T be a derivation tree of some string v w.r.t. the grammar G and let N_T be the set of nodes of T. An *alignment* of w to T is an assignment A of a fragment of w to every node of T that satisfies the following two conditions:

1. The fragment of the root node spans the whole string w i.e., $A(root_T) = (1, n+1)$, where $root_T$ is the root node of T.
2. The fragment of any non-root node is included in the fragment of its parent i.e., if n' is a child of n, then $A(n')$ is included in $A(n)$.
3. The fragment of any inner node follows the fragment of any of its preceding siblings i.e., if n has children $n_1, \ldots, n_m$, then $A(n_j)$ follows (and possibly overlaps with) $A(n_i)$ for any $i \in \{1, \ldots, j-1\}$.

We denote by $start_A(n) = start(A(n))$ and $end_A(n) = end(A(n))$ the start and the end of the fragment of the node n in the alignment A. We define the measure of distance of an alignment A of the string w to a parse tree T recursively on the structure of T. That is, we define a function $cost_A$ that assigns the alignment cost to every node of the parse tree T of some string $v \in L(G)$. We start in the leaf nodes of the parse tree, where we attempt to identify a position within the assigned fragment of w, that is closest to the output atom of the leaf node, and any possible editing operations that need to be performed on the string v. For any leaf note n, the cost $cost_A(n)$ has to take into account whether the fragment $A(n)$ is empty (deletion of material) or not (identity or substitution of material). The latter case may involve distance comparing atoms, as for example the standard Hamming distance equal to the number of renaming operations necessary to obtain the atom $\bar{a}$ from the atom $\bar{b}$.

For an inner node n with m children $n_1, \ldots, n_m$, the definition of the cost is more involved. First of all, the cost of n must include the cost of all its children.

Additionally, it has to incorporate the possible overlaps and gaps between fragments of any pair of two consecutive nodes. Note that the length of overlap/gap for n_i and n_{i+1} is equal to $|end_A(n_i) - start_A(n_{i+1})|$. Also, the cost needs to incorporate the possible left margin between the fragment of the first child n_1 and the fragment of its parent n as well as the right margin between the fragment of the last child n_m and the fragment of its parent. Altogether, we obtain the following formula:

$$cost_A(n) = (start_A(n_1) - start_A(n)) + (end_A(n) - end_A(n_m)) + \\ + \sum_{i=1}^{m-1} |end_A(n_i) - start_A(n_{i+1})| + \sum_{i=1}^{m} cost_A(n_i),$$

possibly with weights on the different cost contributions. Now, the *alignment distance* between a grammar G and a string w is the minimum cost of an alignment of w to a parse tree of G (we assume that G has at least one parse tree):

$$dist(G, w) = \min\{cost_A(root_T) \mid A \text{ is an alignment of } w \text{ to a parse tree } T \text{ of } G\}.$$

Note that an alignment with the minimal cost needs not be unique just as there is more than one way of transforming the string ab to ba with the standard editing operations of deleting and inserting a character.

3.3 Computational challenge

For a given word w and a given parametric grammar G, we want to construct an *optimal alignment* (which includes a parse tree) that minimizes the alignment distance of w to G. The intractability of this problem follows from the high complexity of a much simpler problem of *membership*: given G and w, check whether $w \in L(G)$:

Theorem 1. *Membership for parametric grammars is NP-complete.*

This can be proven with a reduction from a variant of SAT [14]. It is easy to see that $w \in L(G)$ if and only if $dist(G, w) = 0$. Therefore, even measuring the alignment distance alone is intractable and the task becomes more complex when the construction of an optimal alignment is required. Observe that a given parametric grammar can be converted to an equivalent context-free grammar by grounding the nonterminals i.e., substituting the parameters with all possible values. This procedure may yield a context-free grammar of size exponential in the number of occurrences of parameters. Consequently, the number of occurrences of parameters is one source of complexity and bounding it by a constant renders the membership problem tractable.

3.4 Constructing optimal alignment

We now outline an algorithm that constructs optimal alignment for a given input string w and a given parametric grammar G. The basic data structure we employ is a *link* which represents a node of a parse tree of G aligned to a fragment of

the input w with a given cost. Consequently, a link shall constitute of an start and end position, a cost value, a ground term, and if the term is intermediate, also a set of pointers to other links which capture the structure of the parse tree.

During the computation we maintain a set of links which represent partially constructed parse trees each with an optimal alignment to a fragment of the input tree. Initially, this set contains only links with output terms that correspond to aligning the leaves nodes to the output atoms while deleting a number of adjacent atoms and links with output terms inserted at a specific positions of the input string. Then, iteratively, we saturate the set of links with links with intermediate terms that are obtained from applying production rules of G together with any possible move operations. A collection of pointers to the links that triggered using a production rule is stored in the newly created link, and its start, end, and cost is calculated appropriately from the starts, ends, and costs of those links. This process continues until no further link can be added and at this point we search for a link with the smallest cost with start 0 and end $|w|$ that is labeled with the start nonterminal. The cost of this link is the cost an optimal alignment that can be constructed by following the pointers to child links.

4 Computing alignment distances on Bach inventions

We computed the distance computation on a number of inventions using the grammars described in Section 2. When constructing an optimal alignment, an important computational factor is its distance from the input string (i.e., the cost at the root node of the alignment tree). Essentially, the cost of finding an alignment is exponential in its cost. To render the computation feasible we allow the grammar to additionally specify:

- bounds on the length of the fragment derived from a given rule,
- and bounds on the overall cost of aligning any fragment to a given rule (such as ≤ 2 in G_2).

In our experiments we employ a slightly modified cost function that for moving operations does not penalize gaps between elements but only overlaps.

Figure 3 details the derivation tree found for Bach invention #01 by taking a simplified version of the grammar G_2. The invention #01 is decomposed into three parts (P), each one including full TDI sub-parts, even if the grammar allows to skip some of these parts. The computed derivation succeeds thus in finding this 3-part structure, with relevant patterns, even if the computed derivation is not always identical to the reference analysis. By further adjustments in the grammar, it is possible to make the computed derivation even closer to the reference analysis. However, that is not our goal, but rather we show that a single generic grammar can be used to model several music pieces.

Finding a unique grammar that can parse several real pieces is quite difficult: further research need to be conducted to find constraints that are musically relevant while allowing more flexibility in the grammar. However, we show that the same grammar applied on the first three parts of the invention #03 (Figure 4) predicts almost correctly bounds to these parts and corresponding sub-parts.

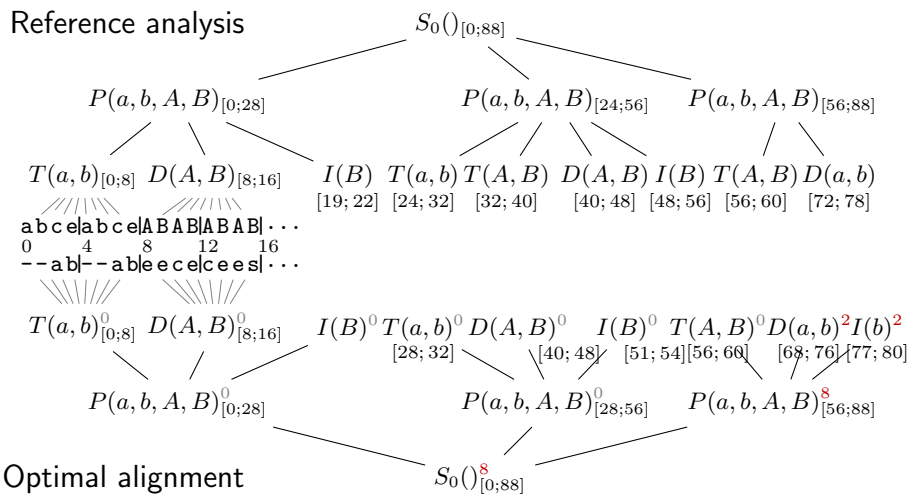

Fig. 3. Decomposition of Invention #01, in C major, with a simplified version of grammar G_2. Each node n is displayed together with, in subscript, the $[start_A(n); end_A(n)]$ values and, in superscript, the alignment distance $cost_A(n)$.

5 Conclusions and Future Work

The proposed formalism of parametric grammars and derivations obtained with optimal alignments offer an attractive way of modeling and identifying the music structure. Our results indicate that our technique is adequate for describing some elements of high-level features of the score.

It should be noted that we started with a paradigmatic analysis giving a first abstract representation of the musical surface. Such an intermediate representation, providing low-level semantics to the music, is here more appropriate that working on a raw stream of notes. Naturally, this kind of patterns could be inferred directly from the musical surface (notes, pitch), or the grammar could produce individual notes. This, however, would add a layer of complexity and its impact needs to be studied further. Also, a number of assumptions and simplifications we have made in our work comes from our intent to use the parametric grammars as a descriptive rather generative model. Possible applications of parametric grammars for music generation need to be explored further.

References

1. M. Chemillier. *Informatique musicale*, chapter Grammaires, automates et musique, pages 195–230. Hermès, 2004.
2. Darrell Conklin. Distinctive patterns in the first movement of Brahms' string quartet in C minor. *Journal of Mathematics and Music*, 4(2):85–92, 2010.
3. David Cope. *Virtual Music: Computer Synthesis of Musical Style*. MIT Press, 2004.

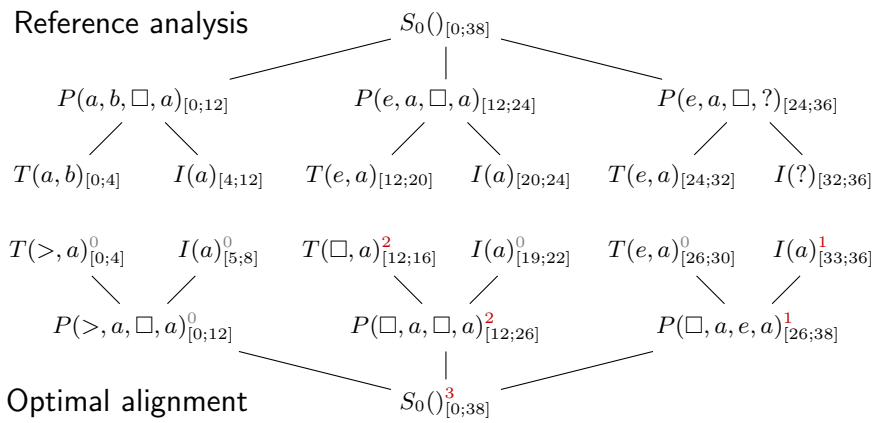

Fig. 4. Decomposition of an extract of Invention #03, in D major.

4. Diana Deutsch and John Feroe. The internal representation of pitch sequences in tonal music. *Psychological Review*, 88(6):503–522, 1981.
5. Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing "a generating theory of tonal music". *J. of New Music Research*, 35(4):249–277, 2006.
6. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2nd edition, 2001.
7. Donald E. Knuth. Semantics of context-free languages. *Mathematical systems theory*, 2(2):127–145, 1968.
8. Olivier Lartillot. Taxonomic categorisation of motivic patterns. *Musicae Scientiae*, 13(1 suppl):25–46, 2009.
9. Fred Lerdahl and Ray S. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983, 1996.
10. Alan Marsden. Schenkerian analysis by computer. *J. of New Music Research*, 39(3):269–289, 2010.
11. David Meredith. A geometric language for representing structure in polyphonic music. In *Int. Society for Music Information Retrieval Conf. (ISMIR 2012)*, 2012.
12. Marcel Mesnage and André Riotte. *Formalisme et modèles musicaux*. 2006.
13. David Neumeyer. The two versions of J. S. Bach's A-minor invention, BWV 784. *Indiana Theory Review*, 4(2):69–99, 1981.
14. T. J. Schaefer. The complexity of satisfiability problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.
15. Heinrich Schenker. *Der freie Satz*. Universal Edition, 1935.
16. Jennifer Shafer. The two-part and three-part inventions of Bach: A mathematical analysis. Honors project, East Texas Baptist University, March 2010.
17. Kirill Sidorov, Andrew Jones, and David Marshall. Music analysis as a smallest grammar problem. In *ISMIR 2014*, 2014.
18. Mark J. Steedman. A generative grammar for jazz chord sequences. *Music Perception*, 2(1):52–77, 1984.
19. J. Sundberg and B. Lindblom. Generative theories in language and music descriptions. *Cognition*, 4:99–122, 1976.
20. Terry Winograd. Linguistics and the computer analysis of tonal harmony. *J. of Music Theory*, 12:2–49, 1948.