# FORMAL MODELING OF STRUCTURAL REPETITION USING TREE COMPRESSION

**Zeng Ren**[1]
EPFL
zeng.ren@epfl.ch

**Yannis Rammos**
EPFL
yannis.rammos@epfl.ch

**Martin Rohrmeier**
EPFL
martin.rohrmeier@epfl.ch

## ABSTRACT

Repetition is central to musical structure as it gives rise both to piece-wise and stylistic coherence. Identifying repetitions in music is computationally not trivial, especially when they are varied or deeply hidden within tree-like structures. Rather than focusing on repetitions of musical events, we propose to pursue repeated structural *relations* between events. More specifically, given a context-free grammar that describes a tonal structure, we aim to computationally identify such relational repetitions within the derivation tree of the grammar. To this end, we first introduce the *Template*, a grammar-generic structure for generating trees that contain structural repetitions. We then approach the discovery of structural repetitions as a search for optimally compressible *Templates* that describe a corpus of pieces in the form of production-rule-labeled trees. To make it tractable, we develop a heuristic, inspired by tree compression algorithms, to approximate the optimally compressible *Templates* of the corpus. After implementing the algorithm in Haskell [1], we apply it to a corpus of jazz harmony trees, where we assess its performance based on the compressibility of the resulting *Templates* and the music-theoretical relevance of the identified repetitions.

## 1. INTRODUCTION

Repetition has been widely recognized as an essential means of establishing effects of coherence in music [1–5]. In Western music, at least, it operates at multiple levels of structure—whether within individual compositions or across different pieces—and takes the shape of musical motifs, themes [6–9], and sectional forms [10–14], among others.

In formalizing repetition in, we need to clarify the *object*, the *means*, and the *scope* of repetition. The *object* specifies the kind of entities being repeated—for instance concrete pitch events or abstract relations between them. By *means* of repetition we refer to processes via which the *objects* are repeated.The *scope* specifies the musical context in which any repetitions are identified.

This paper focuses on *structural repetitions* whose *objects* are not the musical events themselves but rather relations among them. By way of a musical example, consider the first eight bars in the jazz standard "Satin Doll." In common form-theory [10] terms, is sentential; its first half ("presentation") establishes and repeats a "basic idea" one step higher, whereas the second half ("continuation") accelerates the harmonic rhythm towards a closing gesture. [2] A parse tree of the piece (see Figure 1a) based on the jazz harmony grammar by Rohrmeier [15] clarifies the chord dependency and constituency. Here we may observe that the varied repetition of the basic idea in the presentation is reflected in the parallelism between the respective tree components. [3]

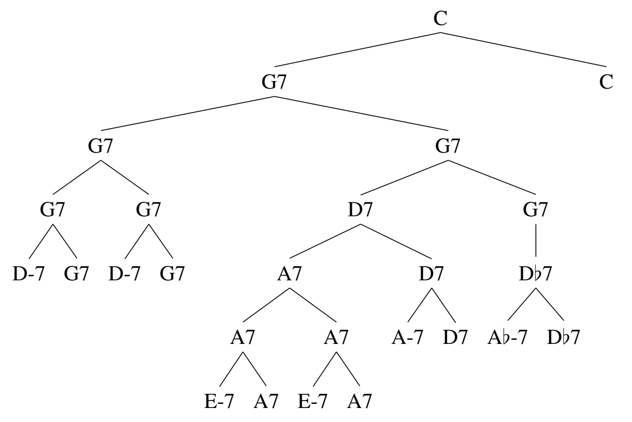
One might think that the tree topology suffices to capture the musical intuition of the parallel structures within this phrase. Further observation reveals that the equality in topology is at most a necessary but not sufficient condition for parallelism. Equality between tree topologies can only express sameness of grouping structure ("constituency"), irrespective of group contents. This is not enough to describe common pattern-like tonal structures such as sequences. The essence of the phenomenon in this example lies also in the equality of the *relations* (edge labels). Listeners familiar with the genre would identify the repeating *objects* as "ii-V"-type motions. To make the relations explicit, we construct a production-rule-labeled tree as shown in Figure 1b. this representation, the parallelism is reflected in the equality of labeled tree components. This notion of repetition, now construed as equality of abstract relations, is crucial in formally capturing parallelism in music.

The *means* of repetition can be informally understood as the coloring present in the rule-labeled tree, which demarcates different rule types. This coloring imposes on the derivation of the piece a repetition constraint that is recursively constrained: indeed, one challenge is to express repetition not just between tree leaves or sub-trees, but also between connected subgraphs of the tree. This generalization would enable us to express deep-level repetitions of tonal frameworks despite non-parallel operations close to

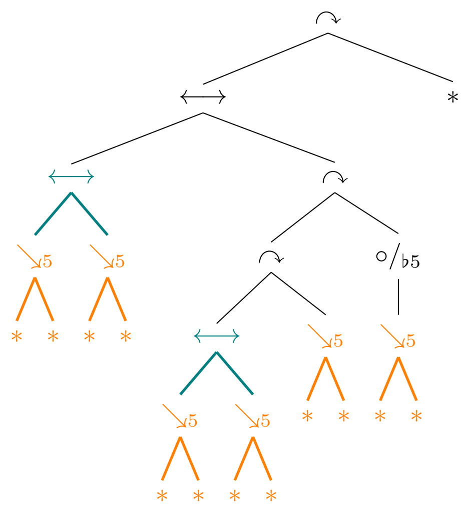
[1] https://github.com/ren-zeng/formal-modeling-of-structural-repetition.git

[2] This hearing becomes obvious after considering the parallelism within the melody (not shown in the figure).

[3] "Tree component" here refers not just to subtrees but also to connected subgraphs within the tree.



(a) Harmony tree of Satin Doll mm. 1-8.



(b) The rule-labeled tree corresponding to the harmonic analysis. "$\curvearrowright$" indicates applied dominant relation. "$\longleftrightarrow$"indicates prolongation. "$\searrow_5$" indicates diatonic descending fifth relation. "$\circ/_{\flat 5}$" indicates tritone substitution.

**Figure 1**: Hierarchical harmonic analysis of "Satin Doll" mm. 1-8 using the jazz harmony grammar [15] under two representations.

the surface (e.g. ornaments). A constructive definition of repetition *means* is provided in section 3.

Finally, we distinguish two kinds of *scopes* for repetition: piece-wise and corpus-wise. This distinction is important for characterizing musical styles. For instance, the "ii-V-I" chord progression in jazz is a recurrent harmonic device **across the corpus**. In contrast, the two descending thirds arranged one descending step apart, which open Beethoven's Fifth Symphony, establish themselves as a motive through specific processes of repetition **within the piece** (and not outside it).

## 2. RELATED WORK

Leaving aside historical texts, a plethora of contemporary theoretical studies have examined the phenomenon of repetition in music [16], as well as the distinction between—borrowing Eugene Narmour's terms [17]—"style structures" [18, 19] and piece-specific "idiostructures" [7, 20].

To computationally model the repetition phenomena in the "Satin Doll" example of section 1, we coordinate two kinds of hierarchical structures: one that explains musical entities in terms of a context-free grammar(CFG), and another that explains the the repetition of grammar rule applications. Hierarchical models of tonal structure are not new [21–25]. The same can be said about the hierarchical understanding of repetition itself such as the String Pattern-Induction Algorithm (SPIA) [26]. Methods for repetition identification have primarily focused on the repeated material itself by searching for exact or inexact successions [27–32] (also see [33] for an overview of this body of research). However, to the best of our knowledge, and outside the sphere of purely music-theoretical contributions, little attention has been paid to computational models of repetition whose objects are relations (generative procedures). Variation, for example, is often understood as a departure from exact repetition by means of ad-hoc or systematic transformations on a concrete entity. In this paper, by constrast, we understand variation as a different elaboration of the same *abstract* entity. Building upon the work by Finkensiep at el. on repetition structure inference [34], we extend its notion of "formal prototypes" to accommodate non-exact repetitions.

In the field of computer science, grammar-based compression algorithms aim to compress data by factoring out repeated information and storing it only once. Grammar-based compression algorithms have been developed and studied both for strings [35] and trees [36–40]. With string-like input data, the Sequitur algorithm [41] encodes a compressed string by constructing a straight-line grammar—a subclass of CFGs—whose language size is equal to one. When the input data have tree or forest form, algorithms such as [35–37, 39, 40] construct a straight-line tree program by iteratively constructing repeated units using "digrams," which assemble a unit of repetition from two adjacent units. In the case of tree patterns, a digram consists of a root plus one of its children. Among the related research, the *TreeRepair* algorithm by Markus Lohrey et al. [37] is relevant as it is specifically concerned with the notion of "deep-level repetition", i.e. of connected graphs within a tree.

In this paper, we formalize repetition patterns as functions operating on generic abstract syntax trees. Using a new approach that is based on tree compression algorithms and formal prototypes, we introduce a model that can discover piece-specific and stylistic patterns from a forest of abstract syntax trees.

## 3. FORMALISM

**Meta-rule.** A *meta-rule* is a list of symbols in $\mathbb{N}^+ \cup \{\_, \star\}$ where "_" denotes a new symbol, "$\star$" denotes the recurrence of the parent symbol ("parent repeat"), and "$n$" denotes the recurrence of the $n$-th argument ("sibling repeat"). In the rest of the paper, we use $\mathcal{M}$ to denote the space of *meta-rules*. Each $m \in \mathcal{M}$ induces a repetition function $rep_m : (X, X^n) \to X^{m+k}$ where the first argument represents the root of a sub-tree and the second represents its children. Here are two examples of the workings of the $rep_m$ function:

$$rep_{\langle \_,\_,1,\star\rangle}(5,[3,1])) = [3,1,3,5]$$
$$rep_{\langle \star,\_,2,\_,\_,4\rangle}(t,[a,b,c]) = [t,a,a,b,c,b].$$

Intuitively, a *meta-rule* encodes the atomic "means" of repetition within a tree structure without specifying the *object* of repetition. [4] In addition, for $m \in \mathcal{M}$, we define $sizeIn(m)$ to be the number of "_" symbols and the $sizeOut(m)$ to be the length of the *meta-rule*.

**Template.** Given a context-free Grammar $G$ where $P_G$ is the set of production rules, we define its corresponding *Template* $\mathcal{T}_G$ constructively using following axioms:

1. (Rule Lifting)
$$\forall f \in P_G. \quad f \in \mathcal{T}_G$$
2. (Composition) [5]
$$\forall f, g \in T_G, i \leq arity(g). \quad (i, g, f) \in \mathcal{T}_G$$
3. (Replication)
$$\forall g \in \mathcal{T}_G, m \in \mathcal{M}, \vec{f} \in \mathcal{T}_G^{sizeIn(m)}. \quad (g, m, \vec{f}) \in \mathcal{T}_G$$

$\mathcal{T}_G$ is effectively a context-free grammar that generates $P_G$-labeled trees. The template is a structured representation of a rule-labeled tree; each template can be mapped to a unique rule-labeled tree but each rule-labeled tree can be assigned to multiple templates. Each rule-labeled tree can be embedded as a template in a trivial way using the axioms *Rule Lifting* and *Replication* with the *meta-rules* containing only "_".

We view the problem of discovering *objects* and *means* of repetition as one of inferring optimally compressible *templates* that generate the given production-rule-labeled trees. Given a collection of rule-labeled trees, we want to parse them as *templates*, so that the total size of the list of *templates* is minimized under memoization. [6]

### 3.1 Atomic parsing operations for $\mathcal{T}_G$

**P-rewrite.** Given a pattern of form $(i, f, x_i)$, referred to as a "digram" in [37], the rewrite procedure operates as depicted in Figure 2 in a post-order fashion, replacing all non-overlapping instances of the pattern within the tree. Through iterative application, *P-rewrite* facilitates the abstraction of a connected subgraph of a tree in the form of a single node. [7]

[4] Visualizing meta-rules as strings of literal symbols, for instance "ABAT", would be more intuitive and readable. Here we opt for a referential representation as it facilitates the computational implementation.

[5] Here the notion of composition extends single-argument function composition to multiple arguments. $g \circ_i f$ denotes the function obtained by passing the output of $f$ to the $i$-th argument of $g$. For *composition* and *replication* we also require the functions involved to have compatible types. The *arity* of a template, informally speaking, is the number of arguments needed to form a complete tree. For a template lifted from a production rule, its *arity* is the number of child symbols. For templates constructed using *composition*, $arity((i, f, g)) = arity(f) + arity(g) - 1$. For templates constructed using *replication*, $arity((g, m, \vec{f})) = \sum_{x \in rep_m(g,\vec{f})} arity(x)$.

[6] This connection to memoization is inspired by [42].

[7] *P-rewrite* mirrors the "replacement step" described in [37]. To the best of our knowledge, this is the first application of this notion in the analysis of musical structure.

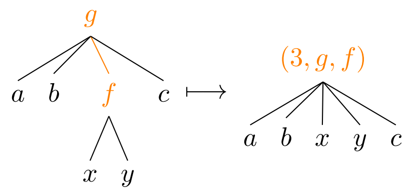


**Figure 2**: A single step of *P-rewrite* using the template $(3, g, f)$ arise from the *composition* axiom.

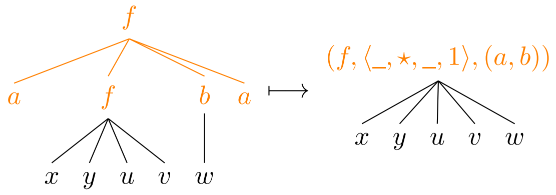


**Figure 3**: A single step of *R-rewrite* using the meta-rule $m = \langle \_, \star, \_, 1\rangle$.

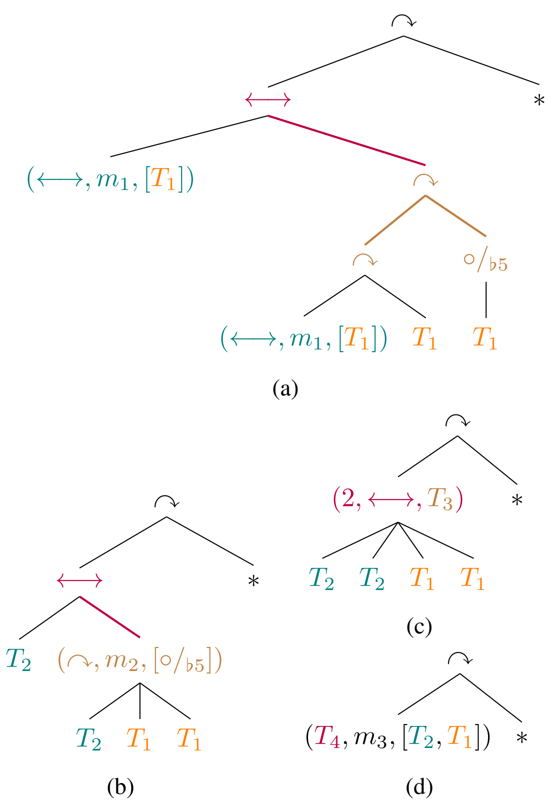
**R-rewrite.** *R-rewrite* is responsible for abstracting *means* of repetition. Given a $m \in \mathcal{M}$, *R-rewrite* applies to the tree whose first-level children are of the form $rep_m(g, \vec{f})$ for some $g$ and $\vec{f}$. For example if $m = \langle \_, \star, \_, 1\rangle$, then the rewrite procedure is the operation shown in Figure (3). *R-rewrite* corresponds to the *replication* case of $\mathcal{T}_G$.

Figure 4 demonstrates the relationship between a rule-labeled tree and the template that generates it. Note that at the most abstract level (Figure 4d) the template shows that the entire dominant region of the *Satin Doll* theme follows a "AABB" repetition pattern (indicated by the *meta-rule* $m_3 = \langle \_, 1, \_, 2\rangle$) where "A" and "B" are templates $T_2$ and $T_1$ respectively. It is worth noting that the *objects* of this particular repetition pattern do not appear at the same structural level in the original rule-labeled tree. In general, the *Template* formalism allows us to coordinate tonal structure and repetition structure in a single hierarchical framework. It is the *Composition* axiom that makes this possible, since it can abstractly represent a connected graph of a tree as a single node. This mechanism is related to tree adjunction and substitution in Tree Adjoining Grammar (TAG) [43, 44]. [8]

## 4. ALGORITHMS

The algorithm draws insights from [37] and [34], in particular their methods for tree and repetition pattern extraction. Departing from [37], our proposed algorithm introduces an additional replacement (*R-rewrite*) step to summarize repetition configurations. In comparison to [34], which operates with strings rather than trees, we introduce a mechanism that handles structural variations, and also derive *meta-rules* from data instead of prescribing a collection thereof. Furthermore, our algorithm differentiates

[8] A *template* of the form $(i, g, f)$ simulates tree adjunction when $g$ is a non-trivial template, and tree substitution when $f$ is a non-trivial template.



$$
\begin{aligned}
T_1 &:= (\searrow_5, m_1, *) & m_1 &= \langle \_, 1\rangle\\
T_2 &:= (\longleftrightarrow, m_1, [T_1]) & &\\
T_3 &:= (\curvearrowright, m_2, [\circ/\flat 5]) & m_2 &= \langle \star, \_\rangle\\
T_4 &:= (2, \longleftrightarrow, T_3) & m_3 &= \langle \_, 1, \_, 2\rangle
\end{aligned}
$$

(e)

**Figure 4**: One possible parse of the rule-labeled tree in Figure 1b. Figures (a-d) demonstrate parsing steps (in bottom-up direction) using P and R rewrites, while figure (e) shows the definition of binded variables, which are colored accordingly.

repetition patterns by their scopes.

The algorithm works by incrementally constructing a table of mined patterns, whether *templates* or *meta-rules*. Its shape is identical to that of Table 1. Given a forest of rule-labeled trees, a pattern is *global* if it occurs at least in two trees, and *local* if it occurs at least twice in a single tree (but not in any other tree).

### 4.1 The Compression Algorithms

The goal is to find the minimal encoding achievable through a series of *P-Rewrite* and *R-Rewrite* operations. An exact solution would require us to try out all the possible rewrite steps (including all possible choices of patterns or meta-rules to write on), with each rewrite generating a new state dependent on the previous state of the program. Even with dynamic programming techniques, such an approach would be computationally intractable. To tame the computational complexity of the optimization problem, we define two 'greedy' heuristics that help find rewrite candidates: a Local Compression for single trees (Algorithm 1) and a Global Compression for forests (Algorithm 2). The complete compression algorithm consists in an iteration of Algorithms 1 and 2 until a fixed-point is reached (when the result convergence).

**Algorithm 1** The Local Compression Algorithm (a Single Step)

```
1: Input
        $t$ : Rule-labeled tree
        $dP$ : Dictionary from symbols to templates
        $dM$ : Dictionary from symbols to meta-rules
2: function $compress_L(t, dP, dM)$
3:     $(o_P, o_r) \leftarrow occurrence_L(t)$
4:     $c \leftarrow bestCandidate(o_p, o_r)$
5:     if $c = Nothing$ then
6:         return $(t, tP, dM)$
7:     else if $c \in o_p$ then
8:         $t' \leftarrow p\text{-}Rewrite(c, t)$
9:         $dP' \leftarrow update(dP, c)$
10:        $dM' \leftarrow dM$
11:    else if $c \in o_r$ then
12:        $t' \leftarrow r\text{-}Rewrite(c, t)$
13:        $dP' \leftarrow dP$
14:        $dM' \leftarrow update(dM, c)$
15:    end if
16:    return $(t', dP', dM')$
17: end function
```

The function $occurance_L$/$occurance_G$ constructs a dictionary of all the patterns and their locations in the tree/forest for potential rewrite. The function $bestCandidate$ in both algorithms is defined by comparing the *net memory savings* of the rewrite.[9] Given a tree $t$, the local frequency of a *composition template* of the form $p = (i, g, f)$, $Freq^P_L(t, p)$, is equal to the maximal non-overlapping occurrences of the pattern within a tree. Its global frequency within a forest $T$, denoted as $Freq^P_G(T, p)$ is the number of pieces where it is present; if it occurs multiple times in a tree of the forest, it still contributes 1 to the global frequency.

The local frequency of $m \in \mathcal{M}$, $Freq^R_L(m)$ is the sum of all local frequencies of the patterns that match a *replication template* $(g, m, \vec{f})$ within a tree.[10] The global frequency $Freq^R_G(T, m)$ simply counts the number of pieces where it occurs.

$$Freq^R_L(t, m) = \sum_{g \in label(t)} Freq^P_L((g, m, children(g))) \tag{1}$$

[9] A net memory saving of a pattern defined as its unit memory saving multiplied by its number of occurrence minus the storage cost. If a pattern is already in the dictionary, then the storage cost is zero.

[10] Similarly with the "non-overlapping" constraint of derivation patterns, we do not count re-occurrences of a meta-rule in a node if the same meta-rule also occurs in any of its children on a repeating position.

**Algorithm 2** The Global Compression Algorithm (a single step)

```
1: Input
        dE : Dictionary from piece-id to $(t, dP, dM)$
        dP_G : Dictionary from symbols to global templates
        dM_G : Dictionary from symbols to global meta-rules
2: function $compress_G(dE, dP_G, dM_G)$
3:     $(d^o_P, d^o_M) \leftarrow occurrence_G(dE)$
4:     $c \leftarrow bestCandidate(d^o_P, d^o_M)$
5:     if $c = Nothing$ then
6:         $dE' \leftarrow compress_L$ over $dE$, only update
7:         local tables if the rule is not in $dP_G$ or $dM_G$
8:         $dP'_G \leftarrow dP_G$
9:         $dM'_G \leftarrow dM_G$
10:    else if $c \in d^o_p$ then
11:        $dE' \leftarrow$ apply p-Rewrite$(c)$ over trees in $dE$
12:        $dP'_G \leftarrow update(dP_G, c)$
13:        $dM'_G \leftarrow dM_G$
14:    else if $c \in d^o_M$ then
15:        $dE' \leftarrow$ apply r-Rewrite$(c)$ over trees in $dE$
16:        $dP'_G \leftarrow dP_G$
17:        $dM'_G \leftarrow update(dM_G, c)$
18:    end if
19:    return $(dE', dP', dM')$
20: end function
```

As an example, in the tree in Fig 1b, the *meta-rule* $m_1 = \langle \_, 1\rangle$ occurs six times [11] while the $template\ T_1 = (\searrow_5, m_1, *)$ occurs four times. [12]

The size for a *composition template* $(i, g, f)$ is always 3 when $g$ and $f$ are stored in memory, because we only need two symbols to represent them, and an integer to specify the location at which they are composed. The $unitSave$ of a *composition template* is always 1, as it replaces two nodes in the tree with one, thus decreasing the tree size by 1. The size of a *replication template* $(g, m, \vec{f})$ is $2 + sizeIn(m)$ for similar reason. The $unitSave$ of meta-rule $m$ is $sizeOut(m) - sizeIn(m)$ as the difference represents the number of symbol repeats.

## 5. EXAMPLE APPLICATION: REPETITION MINING ON THE JAZZ HARMONY TREE BANK

To exemplify an application of the proposed model, we turned to the Jazz Harmony Tree Bank dataset [45], which contains 150 pieces with annotated harmony labels. The annotations are in accordance with the formal-grammar trees of [15]. Since our focus is on modeling repeated relations among chords, rather than the chords alone, we need to transform our input from chord-labeled trees to rule-labeled trees in the same fashion as in Figure 1b. To this end, for every node in the tree we match chord labels with abstract production rules of [15]. The resulting rule-labeled trees are ranked trees, [13] which is a required property for the compression algorithms (see [37]).

[11] as opposed to eight times because of the non-overlapping constraint

[12] This is made more explicit by the reduced tree in Figure 4a.

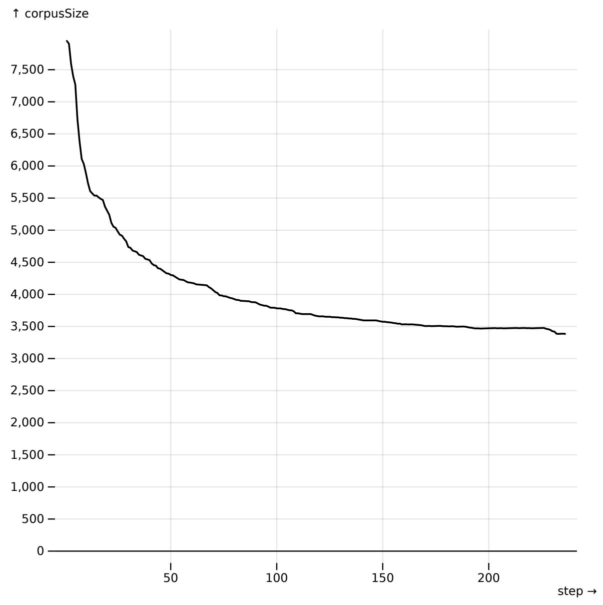


**Figure 5**: Corpus size plotted against the number of global compression steps. The overall reduction in corpus size is from 7948 to 3385.

### 5.1 Results

In global compression, the decrease in tree size in the corpus is partly offset by the expansion of the global-rule table. [14] In initial iterations, this trade-off is highly advantageous, as only few instances of certain patterns in the corpus are needed to offset the cost of storing rules. Thereafter, as shown in Figure 5, the compression size rapidly converges, demonstrating the efficiency of this process. It is worth noting a slight upward trend in the curve towards the end. This is because the algorithm does not mandate a reduction of the corpus size at each step; rather, it extracts patterns as long as they occur twice, whether locally or globally.

Following global compression, each piece is represented in a significantly more condensed format, utilizing the global-rule table. As shown in Figure 6, all pieces are compressed to at least $2/3$ of their original size. In particular, four pieces are compressed to the size of one. [15] These pieces and their changes are the following: "Equinox" (23 to 1), "Mr. P.C." (23 to 1), "Subconscious Lee" (63 to 1), and "Hot House" (63 to 1). Notably, the first two pieces and the last two have the same rule-labeled tree representations respectively, indicating that they are derived in the same manner despite differences in chord labels. They compress to size 1, because their entire "piece" patterns occur at least twice and the algorithm therefore identifies

[13] A ranked tree guarantees that the same symbol has the same arity. In a chord-labeled tree, a chord symbol can occur in both branching node ($arity > 0$) or leaf ($arity = 0$); such a tree is thus not ranked.

[14] The size of a piece is defined by the sum of the size of the template nodes in the tree.

[15] Only two of the pieces whose compressed size is equal to one are visible in the plot due to overlaying.

them as global *templates*. Table 1 summarizes the number of rules obtained after global compression of the corpus. The majority of the rules extracted are global, suggesting many common derivation patterns and meta-rules repeated in multiple pieces. To our surprise, all meta-rules found are global. [16]

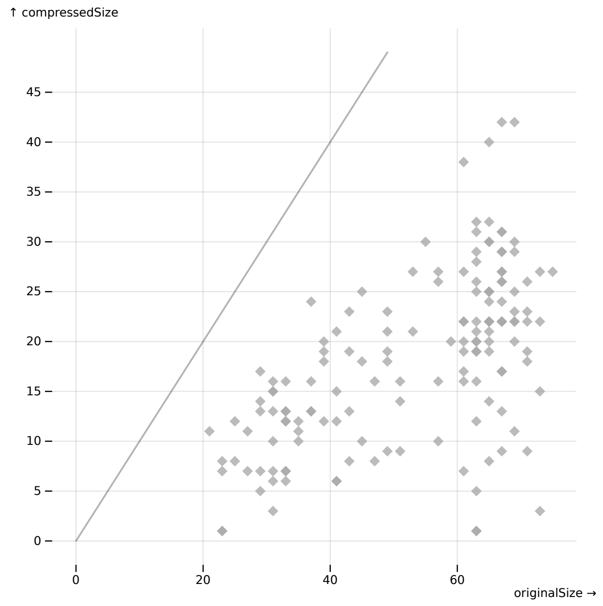


**Figure 6**: Distribution of piece size before ($x$-axis) and after ($y$-axis) global compression. Each dot corresponds to a piece in the corpus. The indicator line represents 1-1 compression rate. Sizes refer to the individual compressed trees alone, not counting the size of global rule tables.

| | Global (stylistic) | Local (piece-specific) |
|---|---|---|
| *Template* | 198 | 20 |
| *Meta-rule* | 36 | 0 |

**Table 1**: The numbers of rules after global compression.

## 6. DISCUSSION AND CONCLUSION

We have presented a formal description and computational model of structural repetition in music, which also accounts for variations. In seeking a compressed representation of a forest of abstracted syntax trees, our goal has been to unveil *what* is actually repeated in a fundamental sense, and *how* entities recur within a specific style. To this end, we proposed a forest compression algorithm based on two rewrite operations, each catering to distinct musical abstractions.

The discovered global *meta-rules* (see Figure 7) include some of the hand-coded *meta-rules* outlined in [34] in line with musical intuition: $\alpha\alpha$ ($M_1$), $\alpha\alpha\beta$ ($M_{64}$) and $\alpha\alpha\beta\alpha$ ($M_{62}$). Also discovered are meta-rules such as $\alpha\beta\alpha\gamma$ ($M_{20}$), akin to (but not necessarily identical with) the "period" in standard contemporary form theory [10], as well as $\alpha\beta\gamma\alpha$ ($M_{65}$), which resembles an expanded ternary structure. *Meta-rules* with parent-child repeats (e.g. $M_4, M_{13}, M_{16}$) emerge quite early in the compression process. The *meta-rule* $\langle\star\rangle$ is the simplest way to nest a pattern. For example, applying it to the template "V region followed by I chord" results in the template "V/V region followed by V chord followed by I chord." [17] We believe such recursive repetitions of the same pattern are, in general, highly meaningful in music. By analyzing the compression rate of the individual pieces after global compression, one could argue that pieces with higher compression rates are generally likely to correspond to more "conventional" expressions of a style. In future research, considering the compression rate of individual pieces could shed light onto their stylistic attributes and patterns of interaction between style and structure.

|  |  |  |
|---|---|---|
| $M_1 = \langle \_, 1\rangle$ | $M_4 = \langle \star, \_\rangle$ | $M_9 = \langle \_, \_, 1, 2\rangle$ |
| $M_{13} = \langle \star\rangle$ | $M_{16} = \langle \_, \star\rangle$ | $M_{17} = \langle \_, 1, 1, 1\rangle$ |
| $M_{20} = \langle \_, \_, 1, \_\rangle$ | $M_{27} = \langle \_, \_, \_, 2\rangle$ | $M_{62} = \langle \_, 1, \_, 1\rangle$ |
| $M_{63} = \langle \_, 1, \_, \_\rangle$ | $M_{64} = \langle \_, 1, \_\rangle$ | $M_{65} = \langle \_, \_, \_, 1\rangle$ |
| $M_{100} = \langle \_, \_, 1, 1\rangle$ | $M_{101} = \langle \_, \_, \_, 3\rangle$ | $M_{148} = \langle \_, \_, 2, \_\rangle$ |

**Figure 7**: The first 15 discovered global *meta-rules* (whose length is less than 5) out of total 36. The index $n$ indicates the $n$-th discovered global pattern, including both *template* and *meta rules*.

We consider the distinction between global and local abstraction meaningful both for music interpretation and its computational representation. Global abstractions enable the creation of more efficient representations of an entire corpus in comparison with intra-piece, local compression. From a music-theoretical perspective, intertextual study is inextricable from the notion of style. For instance, while a ternary form may appear only once within a piece, analysts would still recognize it as a conventional entity because it recurs across the style. Galant schemata [18] can similarly be thought of as collections of stylistic patterns. Form archetypes such as $AABA$ and $ABA$ can likewise be seen as global meta-rules.

By integrating additional types of constraints, the model has the potential, with certain extensions, to express more sophisticated repetitions. For example, Schoenberg's notion of "liquidation" [46] could be recast as the repetition of abstract relations constrained by decreasing elaboration depth. The notion of "fragmentation" [10] could also be modeled as repetition with a constraint on ordering, so that fragments appear only after the initial, integral structure. Our framework could also find use in algorithmic music generation under grammatical constraints. For instance, one could generate a piece in top-down fashion by sampling patterns and meta-rules discovered within a stylistically homogeneous corpus.

[16] We think this is due to the abstract, general nature of *meta-rules*, which makes their recurrence in 150 pieces highly probable.

[17] "Region" here indicates non-terminal symbol while "chord" indicates terminal symbol.

## 7. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 760081 – PMSB. We thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology. We thank the team of the Digital and Cognitive Musicology Lab (DCML), particularly Xinyi Guan, Gabriele Cecchetti and Ran Tamir, for their helpful comments on this paper. We would also like to express our gratitude towards the conference reviewers for their constructive feedbacks.

## 8. REFERENCES

[1] F. Salzer, *Structural hearing: Tonal coherence in music*. Courier Corporation, 1962, vol. 1.

[2] L. M. Zbikowski, "Musical coherence, motive, and categorization," *Music Perception*, vol. 17, no. 1, pp. 5–42, 1999.

[3] ——, *Conceptualizing Music: Cognitive Structure, Theory, and Analysis*. Oxford University Press, 2002.

[4] E. T. Hall and M. T. Pearce, "A model of large-scale thematic structure," *Journal of New Music Research*, vol. 50, no. 3, pp. 220–241, 2021.

[5] E. H. Margulis, *On Repeat: How Music Plays the Mind*. Oxford University Press, 2014.

[6] P. C. Van den Toorn, "What's in a motive? Schoenberg and Schenker reconsidered," *The Journal of Musicology*, vol. 14, no. 3, pp. 370–399, 1996.

[7] D. Beach, "Motivic repetition in Beethoven's Piano Sonata op. 110 part i: The first movement," *Intégral*, pp. 1–29, 1987.

[8] C. Burkhart and H. Schenker, "Schenker's" motivic parallelisms"," *Journal of Music Theory*, vol. 22, no. 2, pp. 145–175, 1978.

[9] A. Cadwallader and W. Pastille, "Schenker's high-level motives," *Journal of Music Theory*, vol. 36, no. 1, pp. 119–148, 1992.

[10] W. E. Caplin, *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*, 1st ed. Oxford Univ. Press, 2001.

[11] Y. Greenberg, *How Sonata Forms: A Bottom-up Approach to Musical Form*. Oxford University Press, 2022.

[12] D. Smyth, "Balanced Interruption" and the Formal Repeat," *Music Theory Spectrum*, vol. 15, no. 1, pp. 76–88, 1993.

[13] R. Ivanovitch, "What's in a theme? On the nature of variation." *Gamut: The Online Journal of the Music Theory Society of the Mid-Atlantic*, vol. 3, no. 1, 2010.

[14] W. Frisch, *Brahms and the principle of developing variation*. University of California Press, 1990, no. 2.

[15] M. Rohrmeier, "The Syntax of Jazz Harmony: Diatonic Tonality, Phrase Structure, and Form," *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, 2020-04-30. [Online]. Available: https://www.ingentaconnect.com/content/10.11116/MTA.7.1.1

[16] J.-J. Nattiez, *Music and Discourse: Toward a Semiology of Music*. Princeton University Press, 1990.

[17] E. Narmour, "Some major theoretical problems concerning the concept of hierarchy in the analysis of tonal music," *Music Perception*, vol. 1, no. 2, pp. 129–199, 1983.

[18] R. Gjerdingen, *Music in the galant style*. OUP USA, 2007.

[19] G. Sanguinetti, *The art of partimento: history, theory, and practice*. OUP USA, 2012.

[20] J. F. Boss, "Schenkerian-Schoenbergian analysis' and hidden repetition in the opening movement of Beethoven's Piano Sonata op. 10, no. 1," *Music Theory Online*, vol. 5, no. 1, 1999.

[21] J. Yust, *Organized Time: Rhythm, Tonality, and Form*, ser. Oxford Studies in Music Theory. Oxford University Press, 2018.

[22] P. Westergaard, *An Introduction to Tonal Theory*. Norton, 1975.

[23] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011-03. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/17459737.2011.573676

[24] ——, "Towards a formalization of musical rhythm." in *ISMIR*, 2020, pp. 621–629.

[25] C. Finkensiep, R. Widdess, and M. A. Rohrmeier, "Modelling the syntax of north indian melodies with a generalized graph grammar," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2019, pp. 462–269.

[26] E. Cambouropoulos, "Towards a general computational theory of musical structure," Ph.D. dissertation, The University of Edinburgh, 1998.

[27] I. Y. Ren, H. V. Koops, A. Volk, and W. Swierstra, "In search of the consensus among musical pattern discovery algorithms," in *Proceedings of the 18th ISMIR*. ISMIR press, 2017, pp. 671–678.

[28] O. Melkonian, I. Y. Ren, W. Swierstra, and A. Volk, “What constitutes a musical pattern?” in *Proceedings of the 7th ACM SIGPLAN International Workshop on functional art, music, modeling, and design*, 2019, pp. 95–105.

[29] A. Laaksonen and K. Lemström, “Transposition and time-warp invariant algorithm for detecting repeated patterns in polyphonic music,” in *Proceedings of the 6th International Conference on Digital Libraries for Musicology*, 2019, pp. 38–42.

[30] J.-L. Hsu, C.-C. Liu, and A. L. Chen, “Discovering nontrivial repeating patterns in music data,” *IEEE Transactions on multimedia*, vol. 3, no. 3, pp. 311–325, 2001.

[31] C. Wang, J. Li, and S. Shi, “N-gram inverted index structures on music data for theme mining and content-based information retrieval,” *Pattern recognition letters*, vol. 27, no. 5, pp. 492–503, 2006.

[32] I. Karydis, A. Nanopoulos, and Y. Manolopoulos, “Finding maximum-length repeating patterns in music databases,” *Multimedia Tools and Applications*, vol. 32, pp. 49–71, 2007.

[33] B. Janssen, W. B. De Haas, A. Volk, and P. Van Kranenburg, “Finding repeated patterns in music: State of knowledge, challenges, perspectives,” in *Sound, Music, and Motion: 10th International Symposium, CMMR 2013, Marseille, France, October 15-18, 2013. Revised Selected Papers 10*. Springer, 2014, pp. 277–297.

[34] C. Finkensiep, M. Haeberle, F. Eisenbrand, M. Neuwirth, and M. A. Rohrmeier, “Repetition-structure inference with formal prototypes,” in *ISMIR 2023 Hybrid Conference*, 2023.

[35] E. Lehman and A. Shelat, “Approximation algorithms for grammar-based compression,” in *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. Citeseer, 2002, pp. 205–212.

[36] T. Akutsu, “A bisection algorithm for grammar-based compression of ordered trees,” *Information Processing Letters*, vol. 110, no. 18-19, pp. 815–820, 2010-09. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020019010002085

[37] M. Lohrey, S. Maneth, and R. Mennicke, “Tree Structure Compression with RePair,” in *2011 Data Compression Conference*. IEEE, 2011-03, pp. 353–362. [Online]. Available: http://ieeexplore.ieee.org/document/5749493/

[38] M. Lohrey, “Grammar-Based Tree Compression,” in *Developments in Language Theory*, I. Potapov, Ed. Springer International Publishing, 2015, vol. 9168, pp. 46–57. [Online]. Available: https://link.springer.com/10.1007/978-3-319-21500-6_3

[39] P. Bille, I. L. Gørtz, G. M. Landau, and O. Weimann, “Tree compression with top trees,” *Information and Computation*, vol. 243, pp. 166–177, 2015-08. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0890540114001643

[40] A. Gascón, M. Lohrey, S. Maneth, C. P. Reh, and K. Sieber, “Grammar-Based Compression of Unranked Trees,” *Theory of Computing Systems*, vol. 64, no. 1, pp. 141–176, 2020-01. [Online]. Available: http://link.springer.com/10.1007/s00224-019-09942-y

[41] C. G. Nevill-Manning and I. H. Witten, “Identifying Hierarchical Structure in Sequences: A linear-time algorithm,” *Journal of Artificial Intelligence Research*, vol. 7, pp. 67–82, 1997-09-01. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10192

[42] T. J. O’Donnell, *Productivity and Reuse in Language: A Theory of Linguistic Computation and Storage*. The MIT Press, 2015.

[43] A. K. Joshi and Y. Schabes, “Tree-adjoining grammars,” in *Handbook of Formal Languages: Volume 3 Beyond Words*. Springer, 1997, pp. 69–123.

[44] A. K. Joshi, “An introduction to tree adjoining grammars,” *Mathematics of language*, vol. 1, pp. 87–115, 1987.

[45] D. Harasim, C. Finkensiep, P. Ericson, T. J. O’Donnell, and M. Rohrmeier, “The jazz harmony treebank,” in *21st ISMIR, Montréal, Canada, October 11-16, 2020*, 2020, pp. 207–215.

[46] A. Schoenberg, G. Strang, and L. Stein, *Fundamentals of Musical Composition*. Faber & Faber, 1999.