MODELING AND INFERRING PROTO-VOICE STRUCTURE IN FREE POLYPHONY

Christoph Finkensiep
École Polytechnique Fédérale de Lausanne
christoph.finkensiep@epfl.ch

Martin Rohrmeier
École Polytechnique Fédérale de Lausanne
martin.rohrmeier@epfl.ch

ABSTRACT

Voice leading is considered to play an important role in the structure of Western tonal music. However, the explicit voice assignment of a piece (if present at all) generally does not reflect all phenomena related to voice leading. Instead, voice-leading phenomena can occur in free textures (e.g., in most keyboard music), or cut across the explicitly notated voices (e.g., through *implicit polyphony* within a single voice). This paper presents a model of *proto-voices*, voice-like structures that encode sequential and vertical relations between notes without the need to assume explicit voices. Proto-voices are constructed by recursive combination of primitive structural operations, such as insertion of neighbor or passing notes, or horizontalization of simultaneous notes. Together, these operations give rise to a grammar-like hierarchical system that can be used to infer the structural fabric of a piece using a chart parsing algorithm. Such a model can serve as a foundation for defining higher-level latent entities (such as harmonies or voice-leading schemata), explicitly linking them to their realizations on the musical surface.

1. INTRODUCTION

A basic observation about tonal structure in music is that notes tend to form vertical and horizontal relations, which are generally not explicit in representations of the musical surface such as a score or a recording. An example of these relations can be seen in Figure 1. The initial line of sixteenth notes in the right hand, for example, forms an arpeggiation of a D-minor chord. A reduction or simplification of the piece might realize this chord as a single vertical entity, but the vertical relation between the notes D5, A4, F4, and D4 is not directly encoded in the score. Similarly, the two A4s of this arpeggiated chord are part of a line that first moves to the neighbor note B♭4 before returning to A4 on the fourth beat of the first bar. Again, this connection is not explicitly represented in the score, much less so in a recording.

Sequential relations between notes are sometimes equated with *voices* [1] that are either explicitly given (e.g. in monophonic melodies or strict polyphony), or inferred through voice separation [2, 3, 4, 5, 6, 7, 8, 9]. However, sequential relations do not always coincide with voices: A single voice can exhibit *implicit polyphony* [10, p. 367][11] (also called implied or latent polyphony), i.e., consist itself of several implied sub-voices. For example, in the upper voice in Figure 1, the notes of the D-minor chord belong to separate voices on a more abstract level. Similarly, sequential connections can go across different voices, such as the A4 moving to B♭4 while the notated voice continues to C♯4.

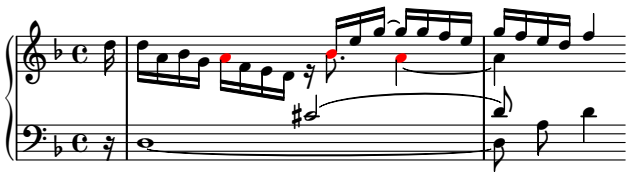


Figure 1: An example of free polyphony in J. S. Bach's Allemande BWV 812 I. Sequential structures (such as the A-B♭-A motion across the first measure) are generally not explicit in the score.

Implicit (or more generally free) polyphony is commonly understood as forming a set of parallel and independent *auditory streams* [12, 13, 1] that are inferred from the musical surface by connecting notes into sequences. The present paper, in contrast, proposes a model of free polyphony that departs from this view in several respects: First, free polyphony is understood as a network of lines that can be connected to each other rather than a set of independent streams. Second, this network is not defined through inference from the surface, but rather explicitly constructed in a generative process that creates the network in successive steps. Inferring this network from a piece is then based on inverting this process, i.e., *parsing* the piece. Third, connections between notes are not based on continuing a stream, but instead follow from *elaboration* of existing structures through fundamental and musically interpretable operations, adopting a top-down view instead of a left-to-right view on voice-leading structure [14, 15]. We name the resulting lines in the network *proto-voices*, since – like voices – they connect notes to sequential lines but cannot be themselves implicitly polyphonic. This paper presents a formal definition of the proto-voice model as a recursive process, and describes a parsing algorithm that can infer the proto-voice structure from a score.

This model does not yet account for other musical aspects such as rhythm and meter, harmony, form, or motivic and thematic material. However, it is intended to further the understanding of polyphonic structure on formal grounds, and could potentially serve as a module in a more complete system for musical analysis.

The idea of modeling free polyphony as a recursively generated network of lines is central to Schenkerian analysis [16]. However, the constructions in Schenkerian analysis are specific to Western Common Practice music and more high-level than the generic operations that give rise to proto-voices. Thus, the proto-voice model can be understood as a formal foundation for describing richer concepts of musical structure (such as the ones appearing in Schenkerian analysis or other analytical frameworks), and it is applicable to a wider range of musical styles that make use of implicit or free polyphony (such as Jazz or melodies in Pop/Rock). [1] However, because of these similar ideas, the model presented here is related to models that formalize sub-systems of Schenkerian analysis [18, 19, 20, 21, 22, 23, 24, 25, 26, 27], and to grammatical models of musical structure in general [14, 15, 28, 29, 17, 30, 31, 32, 33]. While proto-voices inherit some of their concepts, most notably the interval-replacement method developed in [25], modeling the structure of free polyphony has yet been an unsolved problem.

2. THE PROTO-VOICE MODEL

2.1 Constructing Proto-Voices

At the core of the model proposed in this paper are a number of operations that establish primitive and strictly stepwise horizontal relations between notes. These relations include *repetitions*, stepwise ornaments to a note (*neighbor notes*), and notes that fill larger intervals stepwise (*passing notes*). While the notion of a step generally depends on what is considered a step in the respective style, we consider a step to be a diatonic second for the purpose of modeling tonal music in the diatonic tradition.

All of these operations relate notes to one or two reference notes, or *parents*. Following Yust [25], operations with two parents are represented by *edge replacement*: If the two parent notes p_1 and p_2 are connected by an edge $p_1 \rightarrow p_2$, then this edge can be replaced by a child note together with two new edges to the parents: $p_1 \rightarrow c \rightarrow p_2$.

Formally, proto-voices are represented as a graph that contains one vertex per note, one vertex each for the beginning ($\rtimes$) and the end ($\ltimes$) of the piece, and two types of edges: *Regular edges* indicate a sequential connection between two notes (or $\rtimes/\ltimes$) that may be used for elaboration by introducing a repetition or a neighbor of either parent note (or of both if the parents have the same pitch). The interval along a regular edge is always within the range of a step (unless one of its vertices is $\rtimes$ or $\ltimes$), and this property is maintained through the elaboration operations. *Passing edges* indicate connections between two notes with an interval that is larger than a step (introducing a new, subordinate proto-voice). They must be filled with passing notes from either end until only stepwise connections remain.

The generation of a piece starts with the empty piece $\rtimes \rightarrow \ltimes$ and recursively applies one of several elaborations rules. *Single-sided* rules pick a note and insert either a repetition or a neighbor note to its left or right:

$$x \Longrightarrow x' \rightarrow x \quad \texttt{repeat-before} \tag{1}$$
$$x \Longrightarrow x \rightarrow x' \quad \texttt{repeat-after} \tag{2}$$
$$x \Longrightarrow n \rightarrow x \quad \texttt{left-neighbor} \tag{3}$$
$$x \Longrightarrow x \rightarrow n \quad \texttt{right-neighbor} \tag{4}$$

Double-sided rules pick an edge and insert along it one new note and two new edges:

$$\rtimes \rightarrow \ltimes \Longrightarrow \rtimes \rightarrow x \rightarrow \ltimes \quad \texttt{root-note} \tag{5}$$
$$x_1 \rightarrow x_2 \Longrightarrow x_1 \rightarrow x' \rightarrow x_2 \quad \texttt{full-repeat} \tag{6}$$
$$x \rightarrow y \Longrightarrow x \rightarrow y' \rightarrow y \quad \texttt{repeat-before'} \tag{7}$$
$$x \rightarrow y \Longrightarrow x \rightarrow x' \rightarrow y \quad \texttt{repeat-after'} \tag{8}$$
$$x_1 \rightarrow x_2 \Longrightarrow x_1 \rightarrow n \rightarrow x_2 \quad \texttt{full-neighbor} \tag{9}$$

Passing rules, finally, fill passing edges with passing notes from either end until the progression is fully stepwise:

$$x \dashrightarrow y \Longrightarrow x \rightarrow p \dashrightarrow y \quad \texttt{passing-left} \tag{10}$$
$$x \dashrightarrow y \Longrightarrow x \dashrightarrow p \rightarrow y \quad \texttt{passing-right} \tag{11}$$
$$x \dashrightarrow y \Longrightarrow x \rightarrow p \rightarrow y \quad \texttt{passing-final} \tag{12}$$

In these rules, matching letters indicate matching pitches, indices disambiguate parent notes with the same pitch, and apostrophes mark inserted repetitions of parent notes. Neighbor notes n must be a step away from their parents, (disregarding their octaves to allow for octave displacement). Similarly, passing notes p must be a step from the parent(s) they are directly connected to and lie within the interval spanned by both parents. Note that none of these rules produce passing edges, which establish new connections between previously unconnected lines and thus require some additional structure (see Section 2.2. An example proto-voice derivation of the previous example (Figure 1) is shown in Figure 2.

2.2 Temporal Organization

While proto-voices model the sequential organization of notes, they do not specify when notes are simultaneous. On the musical surface, simultaneity of notes is implied by their onsets and durations. However, notes that are temporally displaced on the surface can often be regarded as forming a vertical sonority on a higher level of abstraction, such as the arpeggiated d-minor chord in the beginning of Figure 1. In order to express these latent vertical configurations, simultaneity is modeled through *slices*, segments of a piece in which the same notes sounds. A piece (or a reduction of a piece) is then represented as a sequence

[1] The principle of recursive ornamentation is also used in non-Western styles, such as Indian classical music [17], the model presented here is specifically inspired by Western tonal music. However, some of the formal techniques presented here might also be useful for expressing structural relations specific to other styles.

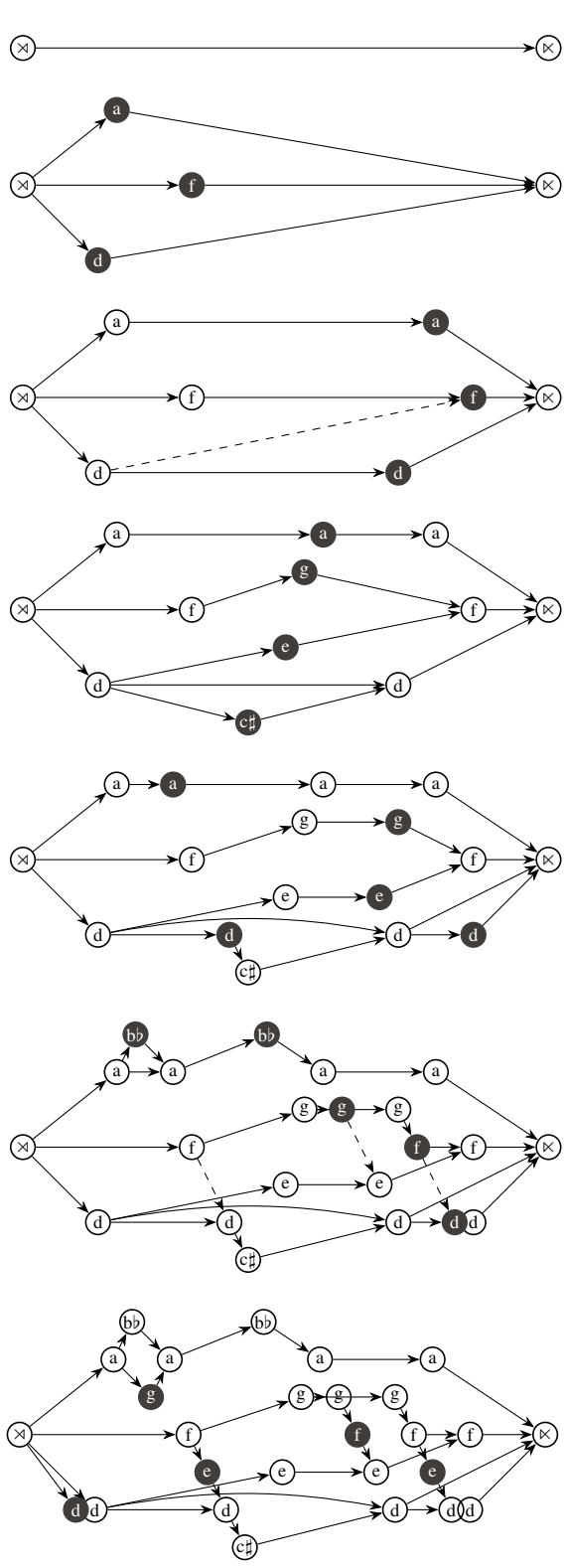
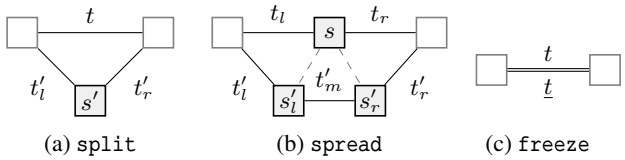


Figure 2: A proto-voice derivation of the notes in Figure 1. The position of a note is chosen to indicate its pitch and onset in the piece. Later derivation steps hide some edges from earlier steps in the interest of readability. Note that each note is shown exactly once here, unlike in the final model, which represents each note once per slice it occurs in. Furthermore, pitches in different octaves have been merged to simplify the graph.



(a) `split` (b) `spread` (c) `freeze`

Figure 3: The three operations on outer structure. The slices and transitions to be elaborated are shown at the top while the lower part shows the generated structure.

of slices. Notes that are simultaneous with several non-simultaneous notes (such as the bass note D in Figure 1) are split among the corresponding slices but remain connected by edges, thus ensuring that a surface note is generated through a single generation process.

Proto-voices are integrated into the slice structure by attaching their edges to the *transitions* between two slices. Note that transitions can only contain edges that connect notes in the slices adjacent to the transition. Long-distance edges are thus represented in latent transitions, i.e. transitions in a reduction of the piece. As a consequence, edges "vanish" in a well-defined manner during the generation process, namely whenever a transition is replaced through one of the generative operations. Since slices and transitions contain notes and edges, respectively, we call the slices and transitions *outer structure*, and the notes and edges *inner structure*.

Formally, a slice s is defined as a multiset (or bag) of pitches. A transition $t = (s_l, e, s_r)$ relates two slices s_l and s_r and a configuration of edges $e = (e_{\text{reg}}, e_{\text{pass}})$, which in turn consists of a set of regular edges e_{reg} (which must be used at least once by a subsequent operation) and a multiset of passing edges (which must be used exactly once). [2]

Outer structure is transformed by three operations: A `split` (Figure 3a) is a rule of the form

$$t \longrightarrow t'_l \; s' \; t'_r \tag{13}$$

that replaces a transition t by inserting a new slice s' and two new transitions t'_l and t'_r. During this operation, each edge in the transition and each note in an adjacent slice can be elaborated by one or more inner operations. The resulting edges can either be discarded, or kept to form the new edges of t'_l and t'_r. As a result, each transition only contains edges that will be used subsequently.

A horizontalization, or `spread` (Figure 3b) has the form

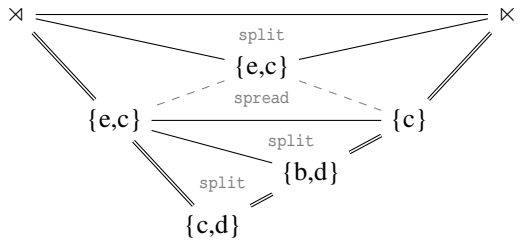
$$t_l \; s \; t_r \longrightarrow t'_l \; s'_l \; t'_m \; s'_r \; t'_r, \tag{14}$$

and replaces a slice s by distributing its notes to two child slices s'_l and s'_r. This way, a latent vertical configuration of notes can be sequentialized. In order to simplify parsing, a restriction is made on this distribution: At least one side must inherit all instances of a specific pitch, while the other may inherit fewer instances, i.e.,

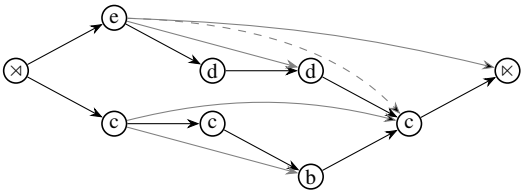
$$p^k \in s \quad \Rightarrow \quad p^k \in s'_l \qquad p^{k-m} \in s'_r \quad \text{or} \tag{15}$$

$$p^k \in s \quad \Rightarrow \quad p^{k-m} \in s'_l \quad p^k \in s'_r, \tag{16}$$

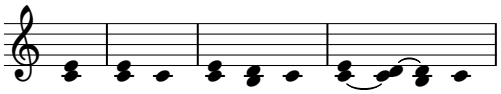
[2] Passing edges are treated differently to avoid filling a single passing edge several times.



(a) Outer structure (slices and transitions).



(b) Inner structure (notes and edges).



(c) The derivation steps in abstracted Western notation.

Figure 4: An example derivation of a short cadential phrase. In each `split` operation, the edges of the elaborated transition (grey in (b)) are replaced using inner elaboration operations. The passing edge from e to c is introduced during the `spread` of the top-level {e, c} slice.

where k denotes the number of occurrences of pitch p in s, and $0 \leq m \leq k$. This way, the s can always be inferred deterministically from s'_l and s'_r by taking for each pitch the maximum number of occurrences in s'_l or s'_r.

In the process of a `spread`, passing edges may be introduced between arbitrary pairs of notes, and regular edges may be introduced between notes with the same pitch. This way, the introduction of passing edges becomes a local operation that is guaranteed to respect the temporal order of notes. Since all edges in a transition must be used, a `spread` is only allowed when no edges from the parent transitions t_l and t_r are lost by moving notes to the opposite side. While this operation does not change the contents of t_l and t_r, it replaces s with s'_l and s'_r respectively, which makes this operation context-sensitive.

Finally, a `freeze` (Figure 3c) marks a transition as terminal, stopping the generation process for this transition:

$$t \longrightarrow \underline{t}. \tag{17}$$

It is only allowed when the transition contains only repetition edges, which are turned into ties, creating notes that span several surface slices.

An example derivation using these three operations can be seen in Figure 4. We use a notation similar to the maximal outerplanar graphs (MOPs) introduced in [25], with the root transition on top, surface of the piece on the bottom, and rule applications indicated by polygons. However, since the derivations here contain latent slices that do not occur on the surface, these derivation graphs are not outerplanar.

3. A PARSING ALGORITHM FOR PROTO-VOICES

3.1 Representing Derivations

The parsing algorithm for the proto-voice model produces a set of possible derivations of the input score. Such a derivation can be represented as a list of rule applications in *leftmost derivation* order. This representation is known from context-free grammars: the result of the derivation is obtained by applying each rule in the list to the leftmost non-terminal symbol of the current sequence. This is possible because the derivation below each non-terminal of a string is independent from the derivations below all other non-terminals of the string. In the proto-voice grammar, this independence property does not hold, because the context-sensitive operation `spread` can link two otherwise independent transitions (and all their ancestors). However, the idea of a leftmost derivation can still be applied here.

The maximal left-hand side of a single rule consists of two transitions. Thus, instead of the leftmost non-terminal, we consider the two leftmost non-terminal transitions as the context for each rule application. Freezing the left of the two transitions moves the context to the right. A `spread` consumes both transitions of the context and pushes its children onto the list of open transitions. In order to allow the right parent of a `spread` to be the result of a `split`, `splits` can be applied to either the left or the right transition of the current context. However, in order to disambiguate the derivation order, we restrict right `splits` to always happen *after* left `splits` or `freezes`. If only a single transition is left, then only a `split` or `freeze` can be performed. Thus, the derivation shown in Figure 4a can be unambiguously described as the leftmost derivation `split`, `spread`, `freeze`, `split-left`, `split-left`, `freeze`, `freeze`, `freeze`, `freeze`.

Under these restrictions, certain configurations are not possible. In particular, the right parent transition of a `spread` cannot be the left child transition of another `spread`. However, this *outer* configuration is equivalent – with respect to the resulting *inner* structure – to another configuration where the two `spreads` are applied in reverse order. Thus, the generative power of the grammar (with respect to proto-voice structure) is not restricted by excluding this non-leftmost configuration.

A similar observation above can be made between `splits` and `spreads`: Whenever a `split` is made *after* a `spread` (i.e. on its left or right child transition), it could as well have been made before the `spread` (generating its left or right parent transition, respectively), generating the same inner structure. Therefore, we can add another restriction on the derivation order that forbids `splits` to be applied to the left or right child transitions of a `spread`, further removing the redundancy between (internally) equivalent derivations.

In a similar fashion, it is possible to reduce the number of derivations further by eliminating redundancy in the internal structure. For example, slices that are exact repetitions of one of their neighbors can be generated in two

ways, either by a `split` that only uses `repeat-*` operations on one side, or by a `spread` that produces identical child slices. Since the latter is required for passing edges, the former case might be excluded as redundant. Similarly, the repeated horizontalization of a vertical configuration can generate the same surface configuration in many different ways, which can be prevented by restricting `spreads` to be strictly left- or right-branching (unless intercepted by a `split`). Both of these restrictions, however, exclude some derivations with slightly different semantics than their permitted counterparts, so it depends on the use cases whether such restrictions are appropriate. [3]

3.2 Parsing

Previous models of hierarchical tonal structure have relied on two approaches to structural inference: Grammar-based models use variants of classical parsing techniques such as chart parsing [28, 27] while MOP-based models work with triangulations of polygons [25, 21]. The proto-voice model can be parsed using a bottom-up chart parsing algorithm that is adapted to account for the context-sensitive `spread` operation. A *transition chart* stores all potential latent transitions, similar to the non-terminal chart in a context-free parser. In addition, a *verticalization chart* stores items that represent the "core" of a `spread`, i.e. the parent slice and the middle transition (including the two child slices). This core is then combined independently with the left and right child transitions, disentangling the two reductions and reducing the combinatorial complexity. [4]

The items in the transition chart are tuples (t, σ, I_l, I_r), consisting of a transition t, a *score* σ, and two IDs I_l and I_r that express combination restrictions on the left and the right of the transition, respectively. By default I_l and I_r have a default value $*$ which indicates that they can combine with other transitions with the default value. The left and the right parent transitions of a `spread`, however, depend on each other through a common child (the `spread` operation itself). They are therefore marked with a special ID on their adjacent sides and can only combine with other transitions with a compatible ID. IDs are based on the *left side* of the verticalization, i.e. its left child slice and its parent slice. The details of the `spread` operation as well as the middle and right child transitions are stored in the item of the right parent transition, while the left parent transition only keeps a reference to the left child transition. This way, combining any pair of compatible left and right parent transitions restores a complete and valid `spread` operation with all its children. While this "trick" reduces complexity by exploiting some properties specific to the proto-voice grammar, it is not known whether it reduces the overall complexity of the parser from exponential to polynomial in the number of input slices.

The score σ of a transition represents the set of leftmost derivations from the transition to the surface it covers. It is computed bottom-up by combining the scores of the transition's children. When two transitions are combined, their scores are combined by concatenating each alternative on the left with each alternative on the right. [5] When parsing a `split` operation, this result is prepended with the `split` itself, which yields the score of the parent transition. The score representing a `spread` operation, however, must be distributed across the two parent transitions. This follows the same scheme as described above: the left parent keeps the score of the left child L; the right parent takes the scores of the right child R, the the middle child M, and the rule application the `spread` itself h. However, since the correct leftmost sequence of operations should apply the scores in the order $hLMR$ the scores of the parent edges are *partial*, and the parser ensures that these fragmented derivations are handled in a way that always restores the correct sequence of derivation steps when recombined. [6]

Algorithm 1 The steps of the parsing algorithm.

$V \leftarrow \{\}$
$T \leftarrow$ unfreeze each input transition
for n from 2 to $|\text{input}| - 1$ **do**
  $V \leftarrow_\cup$ verticalizations of all T_n
  $T \leftarrow_\cup$ left vert. of all $T_n \otimes V_{\leq n}$ and $T_{<n} \otimes V_n$
  $T \leftarrow_\cup$ right vert. of all $V_n \otimes T_{\leq n}$ and $V_{<n} \otimes T_n$
  $T \leftarrow_\cup$ merges of all $T_n \otimes T_{\leq n}$ and $T_{<n} \otimes T_n$
return $T_{\rtimes \to \ltimes}$

The parser fills the chart bottom-up using the algorithm shown in Algorithm 1. Here, *merge* refers to the inverse of a `split`, *left* and *right verticalization* refer to combining a left or right child with a verticalization item, respectively. T_n and V_n refer to the sets of chart items with a surface coverage of n slices, and $\otimes$ creates the pairs of those items that are adjacent (i.e. their connecting slices match with respect to position and content) and have compatible IDs.

The inner structure of each operation is parsed by inverting the operation, computing all possible inputs. For `spread` and `freeze`, this is trivial since their parent elements are unique, if they exist. For `split`, all possible parent transitions are computed that generate every note in s' using all mandatory edges in t'_l and t'_r (and possibly other edges that have been dropped and thus not included in t'_l and t'_r).

A reference implementation of the parser written in Haskell is provided. [7]

[3] For example, with a strictly right-branching model, the expansion of the D-minor chord in Figure 1 must happen from left to right. If it is desired to split the chord first into quarter-note slices and then into eighth-note slices (to respect the metric structure), strict right-branching does not work.

[4] For a given verticalization, instead of considering each pair of left and right transitions ($|L| \cdot |R|$ operations), the left and right transitions can be processed independently ($|L| + |R|$ operations).

[5] In the parser, this operation is represented symbolically, which is more efficient than actually computing all combinations of alternatives.

[6] In particular, since fragmented derivation sets are not always recombined right away, they need to combine with other operations such as `splits` and other `spreads`. The formal details of this are beyond the scope of this paper, but they are documented in the parser implementation.

[7] `https://github.com/DCMLab/protovoices-haskell/tree/ismir2021`

4. DISCUSSION AND CONCLUSION

The proto-voice model is flexible enough to express highly complex configurations of free polyphony. However, this generative power comes at the cost of being highly ambiguous. The suspension sequence in Figure 4, for example, has 131 valid derivations, while the first half measure (including the upbeat) of the Bach example (Figure 1) already has 119,940 derivations. While this flexibility of the model allows analysts to express very subtle interpretative nuances, it also generates the problem that a single piece or excerpt has too many derivations to reasonably compare, and that any non-trivial piece takes far too long to parse exhaustively in practice. The first problem can be solved by introducing a probabilistic variant of the model that weights derivations according to their probability [32, 28]. The second problem might be resolved by a heuristic parser that does not guarantee globally optimal solutions.

There are structural configurations assumed in some theories that require an even higher flexibility than what is provided by the proto-voice model. For example, Schenkerian theory allows the *unfolding* (i.e. horizontalization) of entire progressions (such as the parallel thirds 3-2 and 1-7 in Figure 4) into a single sequence (such as 1-7-3-2(-1)). Such an operation would either require the progression to be represented as a single entity (to which the operation could be applied), or the ability to apply operations to non-entity contexts (similar to how `spread` is applied to two transitions and a slice).

The inner structure and operations of proto-voices are similar to those of MOP-based approaches [25, 21, 24] for monophonic and homophonic sequences. From these, the model inherits the ability to represent double parents and, by extension, lines of notes with a start and a goal. However, proto-voices use these ideas to solve the much more complex problem of free polyphony. The key insight that makes this extension possible is the separation of adjacency on the surface and adjacency in a line of notes, and the explicit representation of line adjacency in the proto-voice graph. In monophonic sequences, surface and line adjacency seem to be the same, but even this assumption does not generally hold: As the example of implied polyphony shows, even monophonic voices can (and generally do) have a polyphonic latent structure. Put bluntly, there is no such thing as a monophonic melody.

The outer structure (and its integration of inner operations) is similar to an approach presented by Marsden [27], that parses single-sided Schenkerian operations based on a grammar on slices. In particular, Marsden's grammar uses *context notes* to model conditions of two-sided operations, which makes the grammar context-sensitive in a very similar way as proto-voices. [8] While Marsden's model does not rely on explicit voices – and thus in principle can parse inputs in free polyphony – it also does not generate voice-like structure among the notes, but rather individual binary dependency relations. A similar point can be made for models working on piano-roll representations such as many neural network approaches [34, 35, 36]: While they can work with freely polyphonic inputs, they generally do not explicitly establish polyphonic *structure* among the notes in the score.

There is, however, a deeper, more philosophical difference between the proto-voice model and the other approaches based on Schenkerian analysis: The proto-voices attempt to isolate and formalize the *structural principles* and *primitives* that give rise to free polyphony, instead of encoding the higher-level concepts and operations of a particular analytical framework. The two structural principles here are *elaboration* and *recursion*, where the former consists of the application of primitives and the latter just arises from the fact that elaboration can be applied to the output of a previous elaboration of the same kind. The structural primitives boil down to essentially two operations: stepwise insertion of notes (in all its variants) and horizontalization of simultaneous elements, which operate with the two basic relations on simultaneity and sequentiality in complementary ways.

These operations are *primitive* for two reasons: First, they provide what can be considered the lowest level of musically meaningful relations. Even more basic representations of music (such as audio or piano-roll representations) do not express musical relations (except incidental simultaneity) explicitly. Second, the basic entities and relations can be combined to express higher-level entities and relations from more specific analytical frameworks, such as different forms of harmonic analysis, Schenkerian analysis, or schema theory. A simple example of such a high-level concept can be seen in Figure 4, which constructs the voice-leading pattern of a *2-3 suspension* in a principled way: first, a progression is generated that moves two voices down in parallel thirds, then another time interval is inserted in which the upper voice moves while the lower voice remains, creating the dissonant second. Similarly, the derivation in Figure 2 explicitly constructs an *initial ascent* [37] from D to F and the harmonic progression $I - V^7 - I$, and describes their relation to the musical surface. The preparatory function of the dominant chord and its dependency on the tonic [15] are even reflected by its notes, which are all ornaments of the following tonic harmony.

The structural principles and primitives postulated by this model are certainly not exhaustive. For one, they do not account for musical parameters such as harmony and key, timbre, or rhythm and meter. Furthermore, there might be additional structural primitives that establish other relations between objects than stepwise motion and simultaneity. Finally, there might be other relevant structural principles, such as abstraction of particular configurations into patterns, or the repetition of complex structures or patterns. However, since principles and primitives are generally orthogonal, the current model can be considered as a module of a more comprehensive model of musical structure.

[8] In [27], this context-sensitiveness is handled by parsing with a context-free parser and then removing inconsistent derivations, while the proto-voice parser only constructs consistent derivations, but this is just an implementation detail.

5. ACKNOWLEDGEMENTS

We thank all members of the Digital and Cognitive Musicology Lab at EPFL (in particular Petter Ericson and Daniel Harasim) as well as the anonymous reviewers for their valuable feedback, and Claude Latour for generously supporting this research. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 760081 – PMSB.

6. REFERENCES

[1] D. Huron. *Voice Leading: The Science Behind a Musical Art*. MIT Press, Sept. 2, 2016. 273 pp.

[2] E. Chew and X. Wu. "Separating Voices in Polyphonic Music: A Contig Mapping Approach". In: *Computer Music Modeling and Retrieval*. Ed. by U. K. Wiil. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 1–20. DOI: 10.1007/978-3-540-31807-1_1.

[3] R. de Valk and T. Weyde. "Deep Neural Networks with Voice Entry Estimation Heuristics for Voice Separation in Symbolic Music Representations". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. 19th International Society for Music Information Retrieval Conference (ISMIR 2018). Paris, France, May 25, 2018.

[4] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé. "Comparing Voice and Stream Segmentation Algorithms". In: *Proceedings of the 16th ISMIR Conference*. International Society for Music Information Retrieval Conference (ISMIR 2015). Malaga, Spain, Oct. 2015, pp. 493–499.

[5] J. Kilian and H. H. Hoos. "Voice Separation-A Local Optimization Approach." In: International Conference on Music Information Retreival. 2002.

[6] P. B. Kirlin and P. E. Utgoff. "VOISE: Learning to Segregate Voices in Explicit and Implicit Polyphony." In: *Proceedings of the Sixth International Conference on Music Information Retreival*. ISMIR. London, 2005, pp. 552–557.

[7] D. Makris, I. Karydis, and E. Cambouropoulos. "VISA3: Refining the Voice Integration/Segregation Algorithm". In: *Proceedings of the Sound and Music Computing Conference 2016*. Hamburg, Germany, 2016.

[8] A. McLeod and M. Steedman. "HMM-Based Voice Separation of MIDI Performance". In: *Journal of New Music Research* 45.1 (Jan. 2, 2016), pp. 17–26. DOI: 10.1080/09298215.2015.1136650.

[9] D. Temperley. "A Unified Probabilistic Model for Polyphonic Music Analysis". In: *Journal of New Music Research* 38.1 (Mar. 1, 2009), pp. 3–18. DOI: 10.1080/09298210902928495.

[10] E. Aldwell and A. Cadwallader. *Harmony and Voice Leading*. Cengage Learning, 2018. 722 pp.

[11] E. Cambouropoulos. "'Voice' Separation: Theoretical, Perceptual and Computational Perspectives". In: Int. Conf. on Music Perception and Cognition (ICMPC). 2006, p. 12.

[12] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT press, 1994.

[13] E. Cambouropoulos. "Voice And Stream: Perceptual And Computational Modeling Of Voice Separation". In: *Music Perception* 26.1 (Sept. 1, 2008), pp. 75–94. DOI: 10.1525/mp.2008.26.1.75.

[14] F. Lerdahl and R. S. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1985.

[15] M. Rohrmeier. "Towards a Generative Syntax of Tonal Harmony". In: *Journal of Mathematics and Music* 5.1 (Mar. 1, 2011), pp. 35–53. DOI: 10.1080/17459737.2011.573676.

[16] H. Schenker. *Free Composition:(Der Freie Satz): Heinrich Schenker; Translated and Edited by Ernst Oster*. Longman, 1979.

[17] C. Finkensiep, R. Widdess, and M. Rohrmeier. "Modelling the Syntax of North Indian Melodies with a Generalized Graph Grammar". In: *Proceedings of the 20th International Society for Music Information Retrieval Conference* (Delft, The Netherlands). Delft, The Netherlands: ISMIR, Nov. 4, 2019, pp. 462–469. DOI: 10.5281/zenodo.3527844.

[18] R. E. Frankel, S. J. Rosenschein, and S. W. Smoliar. "Schenker's Theory of Tonal Music—Its Explication through Computational Processes". In: *International Journal of Man-Machine Studies* 10.2 (Mar. 1, 1978), pp. 121–138. DOI: 10.1016/S0020-7373(78)80008-X.

[19] S. W. Smoliar. "A Computer Aid for Schenkerian Analysis". In: *Proceedings of the 1979 Annual Conference*. ACM '79. New York, NY, USA: ACM, 1979, pp. 110–115. DOI: 10.1145/800177.810043.

[20] J. Rahn. "Logic, Set Theory, Music Theory". In: *College Music Symposium* 19.1 (1979), pp. 114–127.

[21] P. B. Kirlin and P. E. Utgoff. "A Framework for Automated Schenkerian Analysis". In: (2008), p. 6.

[22] P. B. Kirlin and J. Yust. "Analysis of Analysis: Using Machine Learning to Evaluate the Importance of Music Parameters for Schenkerian Analysis". In: *Journal of Mathematics and Music* 10.2 (May 3, 2016), pp. 127–148. DOI: 10.1080/17459737.2016.1209588.

[23] P. B. Kirlin and D. D. Jensen. "Probabilistic Modeling of Hierarchical Music Analysis." In: *Proceedings of the 12th International Society for Music Information Retrieval Conference*. 12th International Society for Music Information Retrieval Conference (ISMIR 2011). 2011, pp. 393–398.

[24] P. B. Kirlin and D. L. Thomas. "Extending a Model of Monophonic Hierarchical Music Analysis to Homophony". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*. Ed. by M. Müller and F. Wiering. 2015, pp. 715–721.

[25] J. D. Yust. "Formal Models of Prolongation". University of Washington, 2006.

[26] A. Marsden. "Representing Melodic Patterns as Networks of Elaborations". In: *Computers and the Humanities* 35.1 (Feb. 1, 2001), pp. 37–54. DOI: 10.1023/A:1002705506386.

[27] A. Marsden. "Schenkerian Analysis by Computer: A Proof of Concept". In: *Journal of New Music Research* 39.3 (Sept. 1, 2010), pp. 269–289. DOI: 10.1080/09298215.2010.503898.

[28] D. Harasim, M. Rohrmeier, and T. J. O'Donnell. "A Generalized Parsing Framework for Generative Models of Harmonic Syntax". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*. Ed. by E. Gómez, X. Hu, E. Humphrey, and E. Benetos. 2018, pp. 152–159.

[29] D. Harasim, T. J. O'Donnell, and M. A. Rohrmeier. "Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony". In: *Proceedings of the 20th ISMIR Conference*. 20th International Society for Music Information Retrieval Conference. CONF. ISMIR, 2019, p. 335. DOI: 10.5281/zenodo.3527812.

[30] É. Gilbert and D. Conklin. "A Probabilistic Context-Free Grammar for Melodic Reduction". In: *International Workshop on Artificial Intelligence and Music, IJCAI-07*. 2007.

[31] M. Granroth-Wilding and M. Steedman. "A Robust Parser-Interpreter for Jazz Chord Sequences". In: *Journal of New Music Research* 43.4 (Oct. 2, 2014), pp. 355–374. DOI: 10.1080/09298215.2014.910532.

[32] S. Abdallah, N. Gold, and A. Marsden. "Analysing Symbolic Music with Probabilistic Grammars". In: *Computational Music Analysis*. Ed. by D. Meredith. Cham: Springer International Publishing, 2016, pp. 157–189. DOI: 10.1007/978-3-319-25931-4_7.

[33] O. Melkonian. "Music as Language: Putting Probabilistic Temporal Graph Grammars to Good Use". In: *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design*. FARM 2019. Berlin, Germany: Association for Computing Machinery, Aug. 23, 2019, pp. 1–10. DOI: 10.1145/3331543.3342576.

[34] W. Chi, P. Kumar, S. Yaddanapudi, S. Rahul, and U. Isik. "Generating Music with a Self-Correcting Non-Chronological Autoregressive Model". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 893–900. DOI: 10.5281/zenodo.4245578.

[35] Z. Wang, D. Wang, Y. Zhang, and G. Xia. "Learning Interpretable Representation for Controllable Polyphonic Music Generation". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 662–669. DOI: 10.5281/zenodo.4245518.

[36] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, G. Xia, and J. Zhao. "PianoTree VAE: Structured Representation Learning for Polyphonic Music". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 368–375. DOI: 10.5281/zenodo.4245446.

[37] A. Cadwallader and D. Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Third Edition. Oxford, New York: Oxford University Press, Mar. 24, 2011. 432 pp.