A FRAMEWORK FOR AUTOMATED SCHENKERIAN ANALYSIS

Phillip B. Kirlin and **Paul E. Utgoff**
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{pkirlin,utgoff}@cs.umass.edu

ABSTRACT

In Schenkerian analysis, one seeks to find structural dependences among the notes of a composition and organize these dependences into a coherent hierarchy that illustrates the function of every note. This type of analysis reveals multiple levels of structure in a composition by constructing a series of simplifications of a piece showing various elaborations and prolongations. We present a framework for solving this problem, called IVI, that uses a state-space search formalism. IVI includes multiple interacting components, including modules for various preliminary analyses (harmonic, melodic, rhythmic, and cadential), identifying and performing reductions, and locating pieces of the *Ursatz*. We describe a number of the algorithms by which IVI forms, stores, and updates its hierarchy of notes, along with details of the *Ursatz*-finding algorithm. We illustrate IVI's functionality on an excerpt from a Schubert piano composition, and also discuss the issues of subproblem interactions and the multiple parsings problem.

1 SCHENKERIAN ANALYSIS

A number of types of music analysis are concerned with "labeling" individual objects in a musical score. In harmonic analysis, labels are assigned to chords and notes corresponding to harmonic function; in contrapuntal voice segregation, labels are assigned to notes indicating voice assignment. A rhythmic analysis may assign different levels of metrical importance to notes. These styles of analysis are similar in that they often describe musical components in isolation, or only in relation to their immediate neighbors on the musical surface.

Structural analysis, on the other hand, emphasizes discovering relationships among notes and chords in a composition, rather than studying individual tones in a vacuum. The word "structure" here refers to "the complete fabric of the composition as established by melody, counterpoint, and harmony in combination" [1].

Schenkerian analysis is the most well-developed type of structural analysis. This type of analysis examines the "interrelationships among melody, counterpoint, and harmony" [1] in a hierarchical manner. Schenker's theory of music allows one to determine which notes in a passage of music are more structurally significant than others. It is important not to confuse "structural significance" with "musical importance;" a musically important note (e.g., crucial for articulating correctly in a performance) can be a very insignificant tone from a structural standpoint. Judgments regarding structural importance result from finding dependences between notes or sets of notes: if a note X derives its musical function or meaning from the presence of another note Y, then X is dependent on Y and Y is deemed more structural than X.

The process of completing a Schenkerian analysis proceeds in a recursive manner. Starting from the musical score of a composition, one may locate any number of structural dependences. After no more can be found, an abstracted score is produced by rearranging or removing the less structural notes. The new abstracted score will reveal new dependences: when their subservient neighbors are moved or eliminated, various structurally important notes in the original score will be deemed less significant to their new neighbors.

This iterative process illustrates Schenker's conception that tonal compositions consist of a "continuum of interrelated structural levels," [1] where each structural level of the music represents that composition at a different level of abstraction. Each successively abstract level expands or prolongs various aspects of the previous one. The most abstract level of the piece is the *background* level. At the other end of the spectrum is the *foreground* level: an analysis at this level usually still contains most of the notes of the score and most closely represents the musical surface. Between these levels is the *middleground* level. While Schenker's own analyses usually only contain these three levels, there can be many levels in between the surface level music and the ultimate background level; rarely are the levels clearly delineated.

Schenker theorized that at the background level, all tonal works could be represented by one of three basic outlines, consisting of a three chord harmonic progression (the *Bassbrechung* or *bass arpeggiation*) supporting a three-, five-, or eight-note descending melodic line (the *Urlinie* or *fundamental line*). Together, these form the *Ursatz* or *fundamen-*

tal structure. The *Ursatz* is a basic structure that appears over most of the length of a single composition, though it often manifests itself at other levels in the structural hierarchy as well. While the *Ursatz* is an important facet of Schenkerian analysis, the idea of structural levels is more encompassing, as the *Ursatz* can be seen as another musical device that may appear at multiple levels of abstraction [2].

2 PREVIOUS WORK AND STATE OF THE ART

The most widely known formalization of musical structure in a hierarchical fashion similar to Schenker's is that of Lerdahl and Jackendoff [8]. The authors attempt to describe the structure of music from a linguistic perspective by providing preference-rule systems that govern various aspects of musical structure. They present a set of rules for grouping of notes in a voice, and another for deducing metrical structure in term of strong beats and weak beats. Their most intriguing contributions here, however, are the preference-rule systems for two recursive reductive systems, one based on "time-span analysis" that is governed by metrical and grouping structure, and another based on "prolongational analysis" that is controlled by the ideas of rising and falling musical tension in a piece. These reductions can be illustrated by trees, where the leaves are the surface-level notes of the piece and higher-level branches represent the more structural tones, or by an equivalent musical depiction as a sequence of staves showing successive levels of the resultant hierarchy of pitches.

Though Lerdahl and Jackendoff frame their discussion in terms of discovering a grammar of music, they acknowledge that their system is incomplete and probably cannot be directly turned into a formal algorithm. This is mainly due to the lack of weightings for the preference rules and that the authors' representations are based primarily on vertical segmentation of the input music as chords, and do not give enough weight to voice-leading considerations. Nevertheless, a number of people have undertaken formalization of Lerdahl and Jackendoff's system, though the results usually require tweaking of parameters. For example, Hirata and Aoyagi [6] present a framework for the representation of musical knowledge in the spirit of Lerdahl and Jackendoff, while Hamanaka et al. [4, 5] describe an implementation that can automatically make reductions according to a set of preference rules, but the weights on the rules must be adjusted by hand for each piece.

Temperley [15] presents models for meter, phrase structure, voice segregation, pitch spelling, harmonic structure, and key structure using a preference rule system very similar to that of Lerdahl and Jackendoff. Temperley, however, went further and was able to use his models in algorithms that "implement Lerdahl and Jackendoff's initial conception." The models and algorithms vary in level of success and sophistication [13]. All of the models developed could be useful in automating Schenkerian analysis, though the concept of hierarchical reductions is not discussed. Temperley later re-developed some of these modules and a number of new ones with a Bayesian statistics slant [16]; here he does present a high-level discussion on how to test or verify Schenker's ideas from a Bayesian viewpoint.

Schenkerian analysis has largely resisted formalization because it was not presented in any sort of formal manner by Schenker himself. Indeed, many textbooks illustrate concepts largely through examples and long paragraphs of prose [1, 3] rather than by giving a recipe for constructing an analysis step-by-step. There have only been three large-scale initiatives in developing a computational procedure for Schenkerian analysis; other relevant work (e.g., Meehan [12]) was undertaken on a smaller scale and was restricted to models, not algorithms. The first large project, the work of Smoliar [14], resulted in a tool used to assist a human in performing an analysis, largely by confirming the validity of reductions; the system did not perform any analysis on its own. The second, by Kassler [7], described a model that took the analysis from the middleground level to the background, but did not work directly with the surface-level musical notes (what one sees in the actual score). The third project in Schenkerian analysis is more current: an undertaking by Marsden [9, 10, 11], which seeks to derive an analysis directly from a MIDI-style music representation. This project, however, is still in its infancy and is focused on calculating all possible reductions using a dynamic programming algorithm and scoring each analysis. For example, it makes no mention of locating the *Ursatz* (a critical step), does not take advantage of other notational information that would be given by a higher-level representation of a score such as MusicXML, and the published research makes no mention of harnessing previous research in voice segregation, key finding, or harmonic analysis to bootstrap the process.

3 THE IVI FRAMEWORK

We now explain the functionality of the IVI framework, a system to perform automated Schenkerian analysis. The IVI name (pronounced like ivy, the plant) was chosen for the I-V-I (tonic-dominant-tonic) harmonic structure that Schenker theorized was at the background level of all tonal compositions. The framework consists of numerous components, which we detail in the following section.

We have chosen MusicXML as the input format for IVI. While the MIDI file format is more widely used for music file storage than MusicXML, the richer score format of MusicXML facilitates the analysis process. A MIDI file gives little information to the user other than pitch and timing information; even such vital clues as the spelling of accidentals is lost. Though hierarchical music analysis is designed to reflect processes in tonal music that are aurally

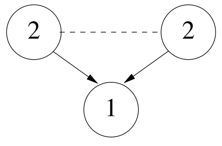

Figure 1. The DAG structure illustrating how the middle note in a neighbor tone musical figure is dependent on the outer tones.

perceived by music theorists and average listeners alike (and so the process could conceivably use MIDI as input), having as much of the printed score as possible available to IVI removes many of the preprocessing steps that otherwise would be inevitable. MusicXML gives us key and time signatures, clefs, beaming information, stem directions, slurs, ornaments, barlines, and written repeats, all of which are potentially useful for Schenkerian analysis, and most of which are not included in a MIDI file.

3.1 Data Structures

The primary data structure that IVI uses to store an analysis (in any state of completeness) is the directed acyclic graph, or DAG. A DAG is an abstract data type consisting of a set of *nodes* (or vertices) and *arcs* (or edges). An arc is a connection between two nodes.

We augment the basic DAG definition to support two different kinds of arcs:

- Regular (directed) arcs: An arc directed from node x to node y specifies that the note(s) represented by y are dependent on those represented by x. That is, a node's children are its dependents.
- Equivalence arcs: These arcs are non-directional. An equivalence arc connecting nodes x and y represents the claim that x and y must be at the same structural level. These arcs are inserted to force nodes to be at the same level.

Note that with equivalence arcs as well as regular arcs between nodes, we can enforce both dependences and structural level equivalences. We show the DAG for a neighbor note idiom in Figure 1, where regular arcs are shown as solid lines, and equivalence arcs are shown as dashed lines.

Initially, the DAG contains one node for each note in the input composition and no arcs; this corresponds to the surface-level information before any analysis has occurred.

3.2 The Informed Search Paradigm

We can formalize Schenkerian analysis as a computational search problem. The search space consists of all possible empty, intermediary, and completed Schenkerian analyses of a piece; since we are using DAGs, this is the space of all possible DAGs over n nodes. This space is prohibitively large for an exhaustive search; Marsden [11] showed that even if we restrict ourselves to DAGs that resemble binary trees, the size of the search space is still factorial in n, the number of notes in the input piece. Therefore, we employ various informed (heuristic) search methods.

A state in the search space consists of an individual DAG, corresponding to a partial Schenkerian analysis. Goal states are completed analyses, where there are only a small number of nodes at the top-most level of the DAG; these nodes will correspond to the tones of the *Ursatz*. The various operators that IVI can apply correspond to either individual Schenkerian reductions, which transform the DAG by adding new nodes and edges, or adding *properties* to the search state. A property is an attribute-value pair that stores additional information about the the state of the analysis not captured by the DAG; the attributes are name-note pairings. The most important property is called `level` and may be attached to any note. This is an integer that represents the importance of that note in the structural hierarchy. Higher numbers represent higher levels of structure. Initially, all notes are assigned a `level` of 1. There is a procedure (detailed later) that updates these values as the analysis progresses. Additional properties correspond to marks an analyst would make on a physical musical score, such as harmonic analysis chord symbols. Since there are myriad property assignments and reductions possible at numerous locations in the analysis, we are investigating using various ranking algorithms to evaluate possible reductions and determine the musical plausibility of each.

Two metrics are needed to perform an intelligent search. If n is a state in the search tree, we need a metric $\hat{g}(n)$ to tell us the distance from the start state (a "blank" analysis) to n, and $\hat{h}(n)$ to tell us the estimated distance from n to a goal state. $\hat{g}$ depends primarily on the number of reductions applied and the aggregate musical plausibility level of those reductions. $\hat{h}$ takes into account more global measurements, such as the plausibility of the current analysis as a whole (rather than individual reductions), the `level`s of each of the most structural notes in the DAG, and the probability of further reductions.

Before the intelligent search begins, however, IVI needs to perform a number of preliminary analyses: a harmonic analysis to determine the initial chord structure, a melodic analysis to determine voice leading, a rhythmic analysis to locate strong beats and weak beats, and a cadential analysis to locate cadences. While we already have basic algorithms in place for these first two analyses, we plan on harnessing the wealth of existing published algorithms for harmonic, melodic, and rhythmic analyses to improve IVI. We have found no existing systems for explicitly finding cadences, however, and so we are currently implementing our own cadence-locating algorithm that will use the results from the other three analysis modules and measurements of harmonic rhythm to identify cadences.

After the preprocessing steps, the search control algorithm begins its work. Once a goal state is found, IVI can save the resulting analysis and backtrack to locate other plausible analyses if desired by the user. We embrace the idea of multiple musically-plausible analyses. Because we are operating with numerical parameters, there will be a single analysis that is deemed the best (or most-likely). Still, much musical knowledge and insight can be gleaned from seeing the "second tier" of analyses.

4 IMPLEMENTATION DETAILS

4.1 Memory Issues

One of the major problems facing a naïve implementation of IVI's search control is that of memory management. Storing a complete DAG and all the properties of all the notes at every search state in the search tree would be infeasible, as the computer would quickly run out of RAM. While we are investigating using beam search to ameliorate this situation, there is a fundamental data structure change we use to lower our memory requirements.

First, we choose to implement the storage of the DAG itself as a collection of properties in a search state. We use properties named "children" and "parents" to store a set of regular arcs indicating structural dependences, and an "equivalent-to" property to store equivalence arcs. By doing this, all local information about a search state is combined into a collection of properties. We cannot store a global set of properties in a central location as we need to support multiple (possibly conflicting) analyses that may interpret the same musical structures in different fashions, leading to different values for the same name-note property attribute.

Because properties are local to a search state, one may be tempted to copy the property set (currently implemented as a hash table) when creating successor states of a particular search state. However, much memory can be saved (with a small sacrifice in time) if we allow properties to be implicitly inherited from a search state's predecessor. When IVI needs to look up a property, it checks the property set of the current search state, and if the property attribute in question is not found, it walks up the search state's chain of preceding states, looking for the appropriate attribute. This inheritance model prevents us from copying properties many times over and wasting memory. We plan on investigating the loss in efficiency due to time spent traversing the search tree, and possibly implementing a compromising solution where properties *are* duplicated, but only at certain states in the search tree.

4.2 Updating the Structural Levels

Updating the `level` of nodes in the DAG must take place after reductions are performed that modify the DAG in some fashion. If one temporarily coalesces nodes connected by equivalence edges into "meta-nodes," then the pure DAG structure is restored while preserving equivalence relations. It is then possible to assign new `level` values by using a dynamic programming algorithm to find the longest path from each node to a node with no parents; these path lengths become the new `level` values.

4.3 Voice Segregation and Searching for the Ursatz

IVI's current voice segregation algorithm works by finding voice-leading connections between individual notes. It prefers stepwise connections between notes, and therefore does very well at identifying cases of implicit polyphony, where voice changes are often signaled by leaps. The procedure also can detect voices being doubled at the octave; instead of creating separate voices for each musical line in the doubling, properties are stored in the current search state indicating the notes are part of a doubling.

Our computational procedure for finding the *Ursatz* in a piece requires we already have the soprano line extracted, which IVI's voice segregation algorithm does for us. IVI first identifies the final most structural cadence (the one with the highest values for `level`) and locates the $\hat{2}$-$\hat{1}$ and V-I pieces. (We use an integer with a circumflex accent to indicate a scale degree.) Next, one must locate possible locations for the primary tone of the *Urlinie*. A method for accomplishing this is to search for candidate primary tones by locating all notes on scale degrees $\hat{3}$, $\hat{5}$, and $\hat{8}$, and selecting the most structural of each scale degree. Now that potential beginning and ending points are determined, the last step is to locate the intermediary tones of the *Urlinie* (not needed for a $\hat{3}$-line). This can be viewed as an optimization procedure: we have a notescape of candidate tones with various levels of structure indicated by the `level` property, and we must pick the descending sequence that maximizes, say, the sum of the levels of structure of each individual tone. This is a variant of the *longest decreasing subsequence problem*, where the items are weighted, and can be solved with dynamic programming. After the *Urlinie* is found, the last remaining note to locate is the initial tone of the bass arpeggiation, which is contained in the initial tonic chord of the composition. Note that as the analysis proceeds, levels of structure may change, so it is possible to run this procedure whenever a new level is created in the analysis.

5 A SHORT EXAMPLE

We illustrate running a few of IVI's implemented components on the first eight measures of Schubert's *Impromptu in B♭ major*, Opus 142, Number 3, depicted in Figure 2. IVI's extraction of the soprano line is shown in Figure 3. Notice how IVI correctly separated the voices in the implied polyphony, leaving only the upper voice in the right hand part. Though it may appear that in measures 5–7, IVI simply took the top note of the right-hand octave doubling, IVI actually detected this doubling (as described earlier) and transfered the register of the soprano voice up an octave.

IVI also correctly locates the manifestation of the *Ursatz* at this level of the *Impromptu*, as displayed in Figure 4. For the primary tone, IVI chose the D6 in measure 5 over the D5 in measure 1 as it prefers later-occurring notes to earlier ones when structural levels cannot be distinguished, as was the case here. (We use scientific pitch notation; C4 is middle C.) We told IVI to look for a $\hat{3}$-line as the *Ursatz*-finding algorithm currently requires specifying the scale degree of the primary tone.

6 FUTURE WORK

6.1 Multiple Parsings

There is no single correct Schenkerian analysis for any given musical composition. Because this type of analysis is based in part on how a listener perceives a piece, different analysts may produce (sometimes significantly) different analyses. For example, a single piece may admit two different types of *Urlinie* (e.g., a $\hat{3}$-line and a $\hat{5}$-line).

IVI must detect when there are multiple musically-valid interpretations of some collection of notes in a piece and consider possible analyses continuing from each of the viewpoints. At the crux of this issue is the problem of conflicting reductions, where two or more reductions can be applied but not independently; applying one precludes the later possibility of applying another. Furthermore, there is also the issue of backtracking if either reduction should lead to a non-musically-valid analysis later on in the process. This is inevitable even when humans perform Schenkerian analysis; reductional decisions can have far-reaching consequences and may occasionally lead to dead ends.

We are considering a number of approaches to the multiple parsings problem. Weighted preference rules are a standard option, having been used with success by Lerdahl and Jackendoff [8], Temperley and Sleator [17], and Temperley [15] in their music analysis formalisms. Our current model for preference rules uses a weighted system for measuring the amount of "evidence" supporting various reductions. The intelligent search framework around which IVI is based can use this evidence metric in its heuristics to choose which analysis paths to follow. The backtracking component of intelligent search gives us a method for handling multiple interpretations and also any dead ends encountered: because the search proceeds down multiple possible paths at once by saving intermediate states in a queue, the algorithm is always currently investigating the state most likely to lead to the most musically-probable analysis.

6.2 Subproblem Interactions and Levels of Influence

Music analysis involves multiple interacting subproblems of various levels of influence. It is impossible to isolate and solve each problem in a vacuum; they are intertwined and cannot (and should not) be separated. This, however, creates a situation of seemingly circular dependences among the problems. For example, doing a complete harmonic analysis without knowing anything about the melodic and contrapuntal aspects of a composition is challenging at best, whereas an analysis of the voice leading of the piece is informed greatly by knowing the underlying chord structure. Music theorists analyze compositions opportunistically, solving the easier cases within a subproblem before tackling the harder ones (within the same subproblem or a different one), as often the easier instances will shed new light on how to solve the harder cases.

Every reduction in IVI has some amount of evidence associated with it. We have already discussed using this evidence measurement to handle the multiple parsings problem; it gives us a method for choosing between multiple non-independent reductions. The evidence for various possible reductions and property assignments, however, is an ever-fluctuating quantity. Each reduction performed has the potential to guide new ones, and so these evidence variables must be updated after every reduction. This suggests a method for improving the performance of the preliminary analyses (melodic, harmonic, rhythmic, and cadential), as one can execute an iterative process by which the next most likely property attribute-value pair is added at every step, and evidence values are updated, possibly causing changes to other property values. This contrasts with other methods for, e.g., harmonic analysis, where often all of the chord labels are assigned in an order that is independent of the actual music being examined.

7 REFERENCES

[1] A. Cadwallader and D. Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Oxford University Press, Oxford, 1998.

[2] A. Forte. Schenker's conception of musical structure. *Journal of Music Theory*, 3(1):1–30, Apr. 1959.

[3] A. Forte and S. E. Gilbert. *Introduction to Schenkerian Analysis*. W. W. Norton and Company, New York, 1982.

[4] M. Hamanaka, K. Hirata, and S. Tojo. ATTA: Automatic time-span tree analyzer based on extended GTTM. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, pages 358–365, 2005.

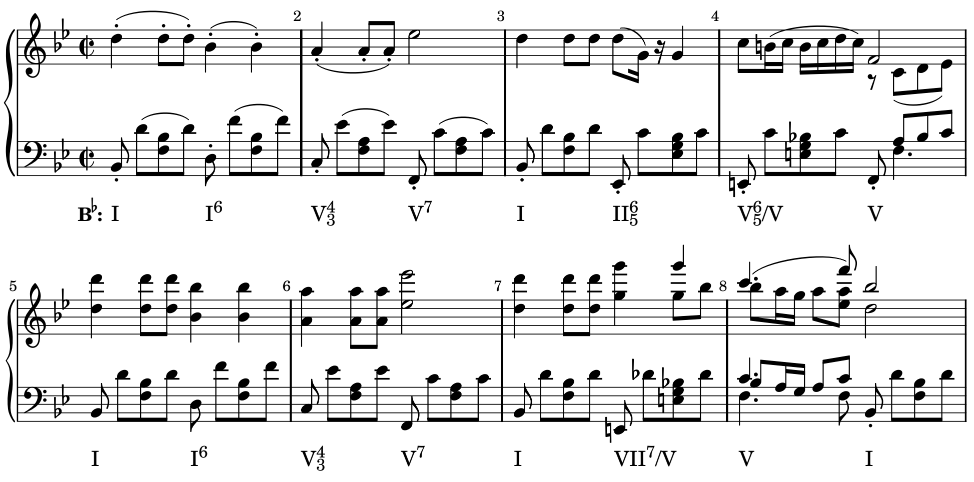

Figure 2. The input score to IVI, with harmonic labels.

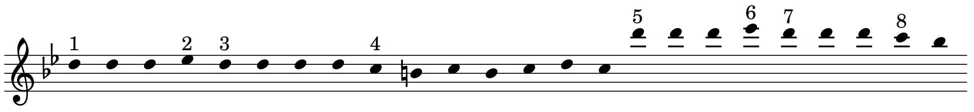

Figure 3. IVI's extraction of the soprano line. (Numbers above notes indicate corresponding measures from Figure 2.)

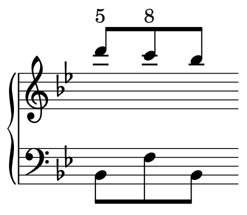

Figure 4. IVI's extraction of the *Ursatz*. (Numbers above notes indicate corresponding measures from Figure 2.)

[5] M. Hamanaka, K. Hirata, and S. Tojo. ATTA: Implementing GTTM on a computer. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 285–286, 2007.

[6] K. Hirata and T. Aoyagi. Computational music representation based on the generative theory of tonal music and the deductive object-oriented database. *Computer Music Journal*, 27(3):73–89, 2003.

[7] M. Kassler. APL applied in music theory. *APL Quote Quad*, 18(2):209–214, 1987. ISSN 0163-6006.

[8] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, Massachusetts, 1983.

[9] A. Marsden. Extending a network-of-elaborations representation to polyphonic music: Schenker and species counterpoint. In *Proceedings of the First Sound and Music Computing Conference*, pages 57–63, 2004.

[10] A. Marsden. Generative structural representation of tonal music. *Journal of New Music Research*, 34(4): 409–428, Dec. 2005.

[11] A. Marsden. Automatic derivation of musical structure: A tool for research on Schenkerian analysis. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 55–58, 2007. Extended Version.

[12] J. R. Meehan. An artificial intelligence approach to tonal music theory. *Computer Music Journal*, 4(2): 60–64, 1980.

[13] D. Meredith. Review of *The Cognition of Basic Musical Structures* by David Temperley. *Musicae Scientiae*, 6(2):287–302, 2002.

[14] S. W. Smoliar. A computer aid for Schenkerian analysis. *Computer Music Journal*, 2(4):41–59, 1980.

[15] D. Temperley. *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, Massachusetts, 2001.

[16] D. Temperley. *Music and Probability*. MIT Press, Cambridge, Massachusetts, 2007.

[17] D. Temperley and D. Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999. ISSN 0148-9267.