AUTOMATIC DERIVATION OF MUSICAL STRUCTURE: A TOOL FOR RESEARCH ON SCHENKERIAN ANALYSIS (EXTENDED VERSION)

Alan Marsden
Lancaster Institute for the Contemporary Arts
Lancaster University, UK

ABSTRACT

This paper describes software to facilitate research on the automatic derivation of hierarchical (Schenkerian) musical structures from a musical surface. Many MIR tasks require information about musical structure, or would perform better if such information were available. Automatic derivation of musical structure faces two significant obstacles. Firstly, the solution space of possible structural analyses of a piece is very large. Secondly, pieces can have more than one valid structural analysis, and there is little firm agreement among music theorists about how to distinguish a good analysis. To circumvent the first of these obstacles, software has been developed which derives a tractable 'matrix' of possibilities from a musical surface (i.e., MIDI-like note-time information). The matrix is somewhat like the intermediate results of a dynamic-programming algorithm, and in a similar way it is possible to extract a particular structural analysis from the matrix by following the appropriate path from the top level to the surface. It therefore provides a tool to facilitate research on the second obstacle by allowing candidate 'goodness' metrics to be incorporated into the software and tested on actual music.

1. THE SIGNIFICANCE OF STRUCTURAL INFORMATION

Many tasks in Music Information Retrieval (MIR) require information about musical structure, or would perform better if such information were available. A prime example is the retrieval of segments which are musically similar. There can be no doubt that, in Classical music, a theme and its variations are similar in some sense, yet the details of both the sound and the actual sequences of notes can be very different. The melody might be heavily ornamented or simplified, and sometimes a completely different melody is used within broadly the same harmonic sequence. Exactly the same applies in the case of jazz improvisation on an existing piece, most readily seen in 'jazz standards'. The similarity in these cases is not in surface features but in the underlying musical structure. While descriptions of structure may be relatively arcane, requiring knowledge of complex music theory, its perception appears to be commonplace: naïve listeners are aware of the similarity between a theme and its variation or an original tune and its rendition by a jazz ensemble.

Software to derive a structural analysis automatically would therefore be a very useful tool in MIR. Software to derive elements of musical structure, such as metre, harmony or grouping, does exist, but not to give a description of the harmonic-melodic pattern of notes. Significant obstacles exist to developing such software, some music-theoretic and some technical, which will be discussed in the following two sections. Thereafter some recently implemented software which goes part-way towards automatic derivation of musical structure, and facilitates systematic research on Schenkerian reduction, is described.[1]

2. STRUCTURE IN MUSIC THEORY

By 'musical structure' I mean a description of the patterns of notes which occur in a piece of music, sufficiently accurate to allow the reconstruction of enough of the actual sequences of notes in the piece to be recognised by most listeners familiar with the original piece. Furthermore, it must contain information about the configurations of notes which is not immediately present in the sequences of notes themselves, and pieces which have different sequences of notes but similar configurations should sound more similar than pieces with equally different sequences of notes but different configurations. For Western tonal music, including Classical music in the period c.1650 to c.1900, plus significant quantities of later music, most film music, popular music and jazz, a number of frameworks for the description of musical structure have been proposed in music theory. (Frameworks proposed for atonal music, some early music and some non-Western music are not widely accepted.) The most widely influential framework in music theory is undoubtedly that proposed by the Austrian theorist Heinrich Schenker [10]. A more systematic theory, very

[1] A fuller description of this project can be found at http://www.lancs.ac.uk/staff/marsdena/research/schenker

different in form but using many of the same ideas, has been proposed by Lerdahl & Jackendoff [6], provoking significant interest among computer scientists.

While the theory of Lerdahl & Jackendoff has the advantage of systematic description, it does not, in my view, give a sufficiently detailed description of a musical structure. It describes a structure of melody plus harmonic support rather than a full contrapuntal structure. I therefore choose to base a structural description on Schenkerian theory. (These issues are discussed more fully in [8], where a computational structural representation based on Schenkerian theory is described.) Furthermore, Schenkerian theory has the advantage of having a large quantity of published analyses which can take the place of a 'ground truth' in the testing of MIR software.

Schenkerian theory describes musical structure in terms of hierarchical levels ('foreground', 'middleground' and 'background' in Schenkerian terms), and analyses are expressed in 'graphs' which demonstrate how a piece of music is constructed by the progressive elaboration of a simple fundamental structure. This is illustrated in Figure 1, which shows an analysis of the first two bars of Mozart's Rondo K.494. (A proper Schenkerian analysis would conflate several of these levels, leaving detail for the reader to infer, and the notation would use noteheads without slurs for higher levels. Figure 1 is intended to be easier to read but to give the same information.) Slurs here are not performance directions but show aspects of the analysis. The slur between A4 and F4 crotchets at the start of the fifth stave, for example, indicates that these two notes join together to form a single chord at the next higher level.

There has been previous study of the possibility of implementing Schenkerian analysis by computer. Kassler [3-5] demonstrated that systematisation of Schenkerian theory was possible and proceeded as far as a system able to derive an analysis from a middleground. Extension of this to derive an analysis from a musical surface has not yet been reported, I suspect in part because of the problem of the size of the solution space, discussed below. More recently Mavromatis & Brown [9] have demonstrated the mathematical possibility of implementing Schenkerian theory as a context-free grammar, but personal communication from Mavromatis indicates that this too has foundered on the problem of the size of the solution space. Gilbert & Conklin [1] get round this by using a probabilistic grammar to derive melodic reductions. Other computer-based work involving aspects of Schenkerian theory has not attempted to generate analyses from actual pieces, (e.g., [11]). Hamanaka, Hirata & Tojo [2] have implemented a system to make reductions according to the theory of Lerdahl & Jackendoff, using their theory of preference rules. However, human intervention is required to adjust parameters to direct reduction towards an acceptable result.

This paper presents the first computer software system which derives quasi-Schenkerian analyses of unconstrained polyphonic pieces of music purely on the basis

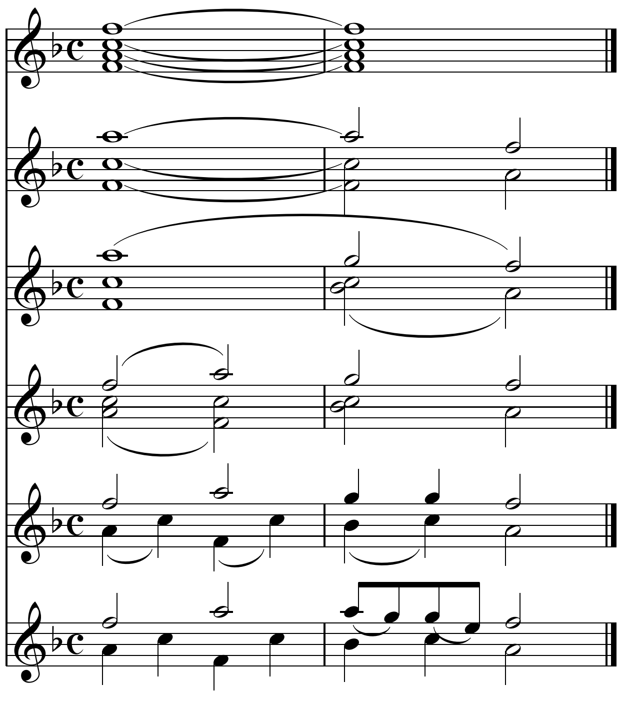
Figure 1. 'Schenkerian' analysis of Mozart K.494

of pitch and time information. However, as is made clear below, there is still considerable work to be done before analyses can be practically derived from full pieces, and before any confidence can be placed in the actual analyses derived. In the first case, the time taken to derive analyses is currently too great, but it is the second issue which is the real area for research. As mentioned above, music theory does not yet supply unequivocal criteria to guide the process of analysis towards a good solution which reflects the structure heard. (Analysts typically rely on their own hearing and musical judgement.) By creating a system which generates sets of analyses, empirical research to determine appropriate criteria is now possible.

3. SIZE OF THE SOLUTION SPACE

In [8], I demonstrate that a Schenkerian analysis of a piece can be represented as a directed acyclic graph which tends towards resembling a binary tree. Each note of the surface of a piece is a terminal node of this graph. The 'roots' are the notes of the highest level reduction. Simultaneous voices tend to be analysed in parallel binary trees, but interactions between voices are represented by links between trees, causing the analysis to become properly a directed graph instead of simply a collection of trees. If we temporarily disregard the constraints which make a graph valid in Schenkerian terms, the number of possible analyses of a piece is at least as many as the number of binary trees possible with n terminal nodes, where n is the maximum number of notes in any voice in the piece. This is the 'Catalan number' C_n: $(2n)!/(n+1)!n!$. Thus we can expect the solution space for Schenkerian analyses of a piece to grow factorially with the size of that piece.

It is very likely that the constraints of Schenkerian theory impose a sufficiently powerful restriction to render this solution space tractable (otherwise how would Schenkerian analyses ever be made?), but in the present state of knowledge we cannot express these constraints with sufficient rigour and confidence to allow the design of a tractable Schenkerian-analysis system. The aim of the research project reported in this paper is to implement a practical tool which facilitates the systematic study of Schenkerian analysis so that these constraints can be discovered and tested. The ultimate objective is to use the results of this research to implement automatic structure-deriving software which completes its task with sufficient efficiency and accuracy for MIR tasks such as segmentation and the discovery of pattern and similarity.

4. SOFTWARE DESIGN

The approach taken in this project is similar to 'dynamic programming': a matrix of local, partial solutions is derived such that a complete solution can be constructed by taking a particular path through the matrix, joining partial solutions to make a complete solution. The surface of a piece is first divided into a sequence of 'segments' such that notes only begin or end at the beginnings or ends of segments. Notes which span several segments are divided into a sequence of notes connected by ties. A segment thus consists of a set of notes (which might be tied to other notes in preceding or following segments) occupying a certain span of time.

4.1. Elaborations

According to tonal theory, only certain elaborations are possible: repetitions, passing notes, appoggiaturas, neighbour notes, suspensions, consonant skips, etc. The precise vocabulary of elaborations will vary according to the musical repertoire. All have a number of characteristics in common, however. (1) One or more 'parent' notes occupying a certain time span at a higher level are replaced by a sequence of two or more 'child' notes at a lower level. (The term 'note' here also admits the possibility of a silent note, i.e., a rest.) (2) An elaboration can depend on the presence of a preceding or following 'context' note (e.g., the preparation of a suspension or the resolution of a neighbour note). (3) The pitch and timing of the child notes are completely determined by the pitch and timing of the parent note(s), plus their tonal, harmonic and metrical context, and any required context note. A candidate set of elaborations is described formally (though within a different framework) in [7] and informally in [8].

Some elaborations can produce a sequence of three or more children (e.g., passing notes) and some can have more than one parent (e.g., an unfolding). To simplify the representation and derivation of reductions, elaborations here are required to have no more than two children and just one parent. Situations where more than one child should occur are represented as more than one elaboration with intermediate stages represented as a special kind of note standing in place of a note sequence. Unfoldings are simply ignored in the current state of the system, since exactly the same reduction arises from combined applications of 'shortening' and 'delay' elaborations whereby a parent note is elaborated to a sequence of a note followed by a rest or a rest followed by a note.

Given any pair of notes at the lower level, and knowledge of the preceding and following notes, it is possible to determine which elaborations could produce those notes and what the parent note would be. Often this depends on a particular tonal, harmonic and metrical context. This information can be attached to the candidate parent as a 'constraint', plus any required preceding or following context note.

4.2. Reduction procedure

The reduction matrix is filled from bottom to top, and right to left (i.e., backwards through the score). This is because knowledge of the presence of preceding and following context notes is required at each step. It is only the suspension elaboration which requires a preceding context, and normally this occurs as a tied note which must therefore be present immediately beforehand in the actual score. (In fact there are occurrences of 'remote' preparations for suspensions in actual music, but these are not common.) Following context notes for neighbour notes or passing notes, however, might be 'remote' and not arise in an immediately following segment until a higher level of the analysis. Proceeding right-to-left ensures that all candidate higher-level following segments are determined before a reduction is made. Preceding surface segments exist already, so any preceding required context notes are already known to be present.

The essential analysis procedure is to take all pairs of consecutive segments and derive from them all possible reductions of that pair of segments. A possible reduction is one in which (1) every note of each segment is a child of some elaboration, (2) the constraints of all elaborations are consistent, and (3) any required context notes are simultaneously present in some possible immediately preceding and following segment.

Two consecutive notes can be reduced to a single parent if they belong to the same musical voice. However, the arrangement of notes in voices is not generally explicit in some music, notably not in piano music. Thus the reduction procedure should perhaps consider all possible arrangement of voices between each pair of segments. To do so, however, would result in extremely large numbers of possibilities to be considered, even with small numbers of notes in each segment. Crossing of voices is possible in music, but extremely rare, so currently the software considers only possibilities with no crossing, and is also has the facility to restrict the joining or splitting of voices (when two or more voices

c1: F5 C5 **or** F5 A4 **or** F5 C5 A4		
	b2: F5 _C5 A4	
b1: G5 C5 **or** E5 C5 **or** G5 E5 C5		
a1: G5 C5_	**a2**: E5 _C5	**a3**: F5 A4

Figure 2. Extract from analytical matrix, covering the last three segments of the example in Figure 3.

proceed to or follow from the same note) to further reduce the number of possibilities to be considered. (The effect of these restrictions is a topic for future research.)

The result of this step is a set of new segments, all occupying a span of time which is the sum of the spans of the two 'child' segments. The number of segments in this set can be large, but it is limited because segments are only distinguished by the notes they contain (plus details of contextual requirements, but these are also limited), and there is only a finite (and relatively small) set of possible notes. The procedure is then applied recursively to all resulting segments until the top level, where segments cover the entire span of the piece. The result is a triangular matrix of sets of segments which constitutes a conflation of all possible reductions of the piece. To derive a complete reduction, one need only select one top-level segment and then recursively select pairs of children, forming a binary tree whose leaves are the segments on the surface of the piece.

The analysis procedure can be explained further by reference to Figure 2, which reflects the analysis of the end of the example shown in Figures 1 & 3. Cells a1 to a3 reflect the last three segments of the example. The segments of cell b1 are derived by finding all possible ways of combining the segments of a1 and a2 so that consecutive notes form permissible progressions whose harmonic and tonal constraints are consistent. The tied notes C5 in a1 and a2 can only combine with each other, but the G5 and E5 can combine in three different ways, resulting in G5, E5 or both G5 and E5 at the level above. Thus cell b1 contains three segments, all of them containing the note C5 while E5 and G5 appear in two each. Cell b2 is derived from combining a2 and a3, and contains the segment which is the only permissible way of combining these notes into a single chord. Cell c1 is derived from combining both a1 with b2 and b1 with a3. There are several possible ways of combining these segments, but they all result in just three segments, all of which contain F5 while C5 and A4 are contained in two each.

The basic size of the matrix (the number of sets of segments) is obviously related to the square of the length of the music analysed, so the space requirement of the reduction algorithm can be expected to be of order $O(n^2)$. However, the number of pairs of spans to be considered, when deriving the new segments for a new longer span, increases at each higher level of the matrix, and the time requirement is of order $O(n^3)$. The real constraint on tractability, however, is the number of segments in each set. The upper limit on this number is 2 raised to the power of the total number of different notes which might make up a segment. This is the number of different notes in the music analysed, which is not (necessarily) related to the length of the music, and furthermore is limited by the number of different notes possible in any piece of music, which is fixed by the instrument(s) on which it is to be played. Thus this does not, in principle, increase the order of complexity of the algorithm. However, the number of possible segments is extremely large. For example, an eighteenth-century piano has 61 notes, and so there are approximately $2.3*10^{18}$ possible combinations of different notes which could appear in segments. Many of these are harmonically impossible and/or impossible to play, but the number of harmonically possible and playable segments is still extremely large. The time taken by the analysis procedure is related to the square of the average number of segments for each span, so a truly tractable analysis procedure depends on keeping this number small. Currently, deriving the matrix of reductions (with no limit on joining and splitting of voices) for a fragment with just 15 segments takes about 5 minutes.

Schenkerian analyses, however, generally consist of just a small number of voices (typically three, four or five). Thus one possible route to making the reduction procedure more efficient is to restrict the size of segment which can be produced by reduction.

Furthermore, although the metre of a piece probably cannot be derived with certainty prior to reduction, certain rhythmic combinations are extremely rare. Thus a further simplification of the reduction procedure restricts reduction to those segments whose durations are related by relatively simple ratios.

The procedure remains extremely time-consuming, however. For example, restricting segments to no more than four notes, disallowing any joining and splitting of voices, and requiring the duration ratios of reductions to have a denominator no greater than four, it still took one hour and forty-five minutes (and 170Mb of heap space) to fill the reductional matrix for a four-bar extract from a Mozart piano sonata. In this case there were several thousand possible segments found for a few cells of the matrix. At the moment segments with the same notes but different constraints are counted as different segments (because they might fit or not fit with others' contexts differently). A development of the procedure currently being tested is instead to count such possibilities just once and keep track of the various alternative constraints. This would reduce the several thousand possibilities to a few hundred. (Further revisions, described below in section 6, should reduce this further.)

5. AN EXAMPLE

Figure 3 shows a result of applying the software to the music example in Figure 1 (Mozart's Rondo, K.494).[1] As indicated above, the software generates a matrix containing a set of possible analyses. The software includes a number of mechanisms for assigning a score to each segment (e.g., the minimum total number of notes in this segment and all its descendents), and a mechanism for pruning the matrix so that only segments with the best score are retained.

One possible way of assigning a score to a segment is to count the minimum number of elaborations required to derive this segment from the surface. (The scoring is a little more sophisticated than a simple count, in that repetitions count for less than neighbour notes, for example.) When this is applied to the matrix of segments arising from analysis of the first two bars of Mozart's rondo, and when only the best-scoring segments are retained, only four possible complete analyses remain. These share the same segments at every point except the first segment of the second-highest level where various combinations of G5, F5, Bb4 and F4 are possible. Figure 3 shows the resulting analysis when the segment with all of these notes is chosen.

The analyses of Figures 1 and 3 do not match, so in that sense the software has failed to derive the correct analysis of this music. On the other hand, there are only two fundamental errors in the analysis of Figure. Firstly, the reduction of the last two chords of the fourth stave produces a bad rhythm in the third stave (the software currently does not take rhythm into account at all). Secondly, the reduction of the first two chords in the third stave produces a bad chord in the second stave (the software currently does not take account of inversions of chords, of harmonic sequence, or of the expectation to start on the tonic). This result can therefore be described as promising.

6. FURTHER RESEARCH

Currently, the reduction procedure is being revised to reduce the number of possible segments recorded in each cell. As described above, this entails a redesign to attach alternative constraints to a set of notes in a single segment rather than recording such cases as alternative segments. The loosest constraints will be used in making the reductional matrix, but tighter constraints applied if the children giving rise to the loosest constraints are subsequently deleted from the matrix.

Additional revisions are to tighten the application of harmonic constraints. Sevenths will be required to resolve properly to make a valid reduction. Harmonic constraints will be passed up from children to parents more tightly also.

Further research will incorporate broader considerations of rhythm and structural norms into scoring mechanisms. These mechanisms will be tested by comparing the resulting generated analyses with actual analyses by Schenker and his pupils. Successful scoring systems will form the basis of mechanisms for pruning bad analyses from early in the derivation process, in the hope of arriving at a reliable structure-derivation system which is sufficiently reliable to form the basis of MIR systems.

[1]Demonstration software, including this example, may be viewed at http://www.lancs.ac.uk/staff/marsdena/schenker

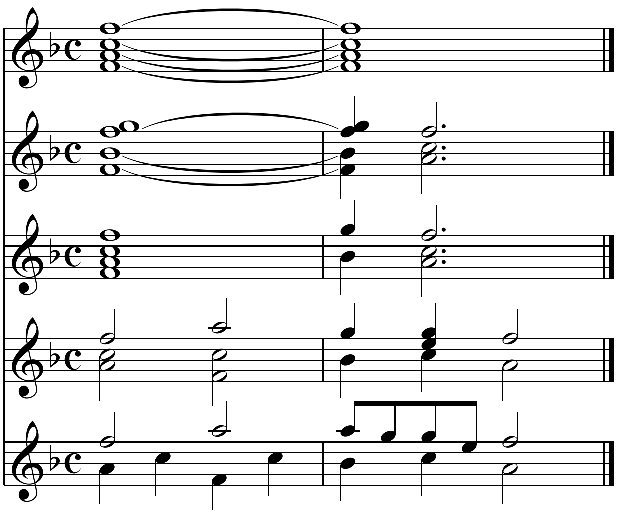

Figure 3. Automatic analysis of Mozart K.494

7. REFERENCES

[1] Gilbert, E., & Conklin, D., "A Probabilistic Context-Free Grammar for Melodic Reduction", International Workshop on Artificial Intelligence and Music, IJCAI-07, Hyderabad, India, 2007.

[2] Hamanaka, M., Hirata, K. & Tojo, S., "ATTA: Automatic Time-Span Tree Analyzer Based on Extended GTTM", in Proceedings of the Sixth International Conference on Music Information Retrieval, ISMIR 2005, 358–365.

[3] Kassler M., Proving Musical Theorems I: The Middleground of Hienrich Schenker's Theory of Tonality (Tech. Rep. No. 103). Sydney, Australia: University of Sydney, School of Physics, Basser Department of Computer Science, 1975.

[4] Kassler M., "Explication of the middleground of Schenker's theory of tonality", *Miscellanea Musicologica: Adelaide Studies in Musicology*, v.9, 72–81, 1977.

[5] Kassler M., "APL applied in music theory", *APL Quote Quad*, v.18, 209–214, 1988.

[6] Lerdahl, F. & Jackendoff R., *A Generative Theory of Tonal Music*, MIT Press, 1983.

[7] Marsden, A. Representing Melodic Patterns as Networks of Elaborations. *Computers and the Humanities*, v.35, 37–54, 2001.

[8] Marsden, A., "Generative Structural Representation of Tonal Music", *Journal of New Music Research*, v.34, 409–428, 2005.

[9] Mavromatis, P., & Brown, M., “Parsing Context-Free Grammars for Music: A Computational Model of Schenkerian Analysis”, Proceedings of the 8th International Conference on Music Perception and Cognition, Evanston, USA, 2004, 414–415.

[10] Schenker, H. *Der frei Satz,* Vienna: Universal Edition, 1935. Published in English as *Free Composition*, translated and edited by E. Oster, Longman, 1979.

[11] Smoliar, S.W. “A Computer Aid for Schenkerian Analysis”, *Computer Music Journal*, v.4, 41–59, 1980.