SOFTWARE FOR SCHENKERIAN ANALYSIS

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

ABSTRACT

Software developed to automate the process of Schenkerian analysis is described. The current state of the art is that moderately good analyses of small extracts can be generated, but more information is required about the criteria by which analysts make decisions among alternative interpretations in the course of analysis. The software described here allows the procedure of reduction to be examined while in process, allowing decision points, and potentially criteria, to become clear.

1. INTRODUCTION

Schenkerian analysis [8] is the most sophisticated and widely used method of explicating the structure of a piece of tonal music at a range of scales, from a sequence of a few notes to entire movements. In providing a method of partitioning the stream of notes which makes up a piece of music, and describing the interrelation and function of elements, it fulfils a role rather like that of grammar for language. Schenkerian theory has many and influential detractors, but its controversial aspects (principally its strong normativity, arising from the chauvinism of its author, such as the insistence that all good pieces of music share a small number of background structures) are not necessarily essential to the usefulness of other aspects. Alternative theories forming a similar role are either related (such as [4]) or no better supported by evidence. Music Theory is stuck in a rut where argument between competing theories is based on, at best, small numbers of example analyses and, at worst, prejudice. Implementation of an analytical theory in computer software allows objective testing, and uncovers areas of underspecification in the theory.

A second potential dividend of computational implementation of Schenkerian analysis is as a basis for software tools which facilitate the manipulation of music at a level between that of notes and entire movements, other than by arbitrarily defined sets of notes or events.

2. STATE OF THE ART

Following in a history of projects spanning more than three decades [2-3], software has recently been developed to make quasi-Schenkerian reductions of short segments (four to eight bars) of music from a representation of the score without the intervention of a human expert [7]. The measure of success for that project was the degree to which the resulting reductions matched analyses made by human experts. While the results were encouraging, the basis of evidence was small. What the project did show clearly was that more information is needed about the criteria by which analysts make judgements, and the process used in analysis. The fundamental problem encountered in the project was that the stated principles of Schenkerian analysis were found to allow vast numbers of alternative analyses of an extract of music, but the principles by which particular alternatives are selected are unclear.

Certain selection principles were established in that project (e.g., avoiding syncopation), but the method used does not readily scale up to investigate more comprehensively because the quantity of suitable available test materials is small, and the method is extremely time consuming. This paper therefore reports a development of that Schenkerian analysis software which allows the process to be observed and probed in the course of deriving an analysis, and allows some intervention from the user. This lays bare places where the software operates inefficiently or makes bad decisions. It can thereby function as a tool for investigation of the process of Schenkerian analysis.

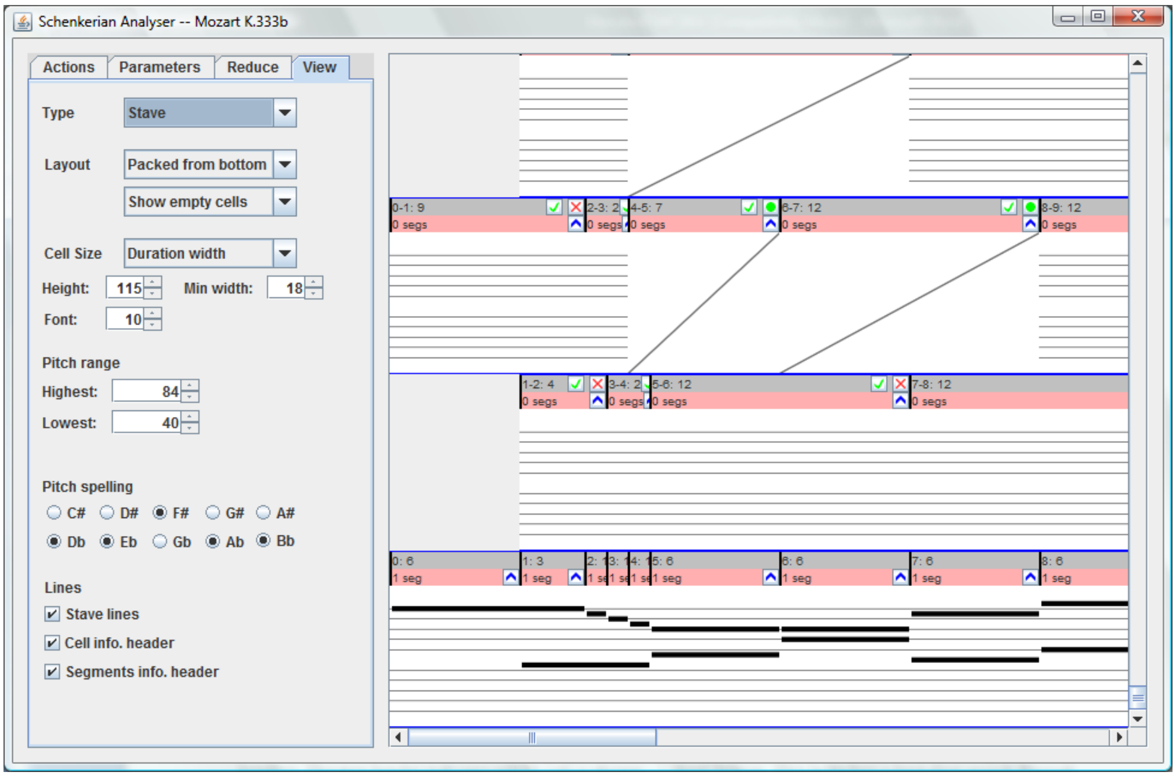
3. BASIC PRINCIPLES

Schenkerian analysis expresses the structure of a piece of music through several layers of reduction. At the lowest level is the ‘surface’ of the piece, represented by the notes in the score. At the highest level is the ‘Ursatz’, an instance of a fundamental structure: I-V-I in the bass with a linear descent to the tonic in the top voice. Each level reduces the level below by replacing sequences of notes in that level with single notes at this level. For example, a pattern C-D-C might be reduced to a single C.

The formalisation implemented here (described in full in [7]) simplifies this so that every reduction is of a pair of consecutive notes (or a note plus a rest) to a single note (or rest). Every reduction which reduces more than two notes to one can be expressed as a set of nested reductions of this binary type.

Reductions are constrained to belong to one of a small number of patterns, such as neighbour-notes, appoggiaturas, etc. Each reduction has harmonic implications (certain pitch classes must belong to the prevailing harmony) and the implications of simultaneous reductions must be mutually consistent.

Some reductions depend on context, meaning that notes of certain pitches must occur immediately beforehand (for a reduction such as a suspension) or afterwards (for a reduction such as an anticipation). A consequence of this manner of representation is that in the case of a reduction such as an anticipation, a note is reduced, counter-intuitively perhaps, with a preceding



Schenkerian Analyser -- Mozart K.333b

Actions | Parameters | Reduce | View

Type: Stave

Layout: Packed from bottom

Show empty cells

Cell Size: Duration width

Height: 115 Min width: 18

Font: 10

Pitch range

Highest: 84

Lowest: 40

Pitch spelling

○ C# ○ D# ● F# ○ G# ○ A#

● Db ● Eb ○ Gb ● Ab ● Bb

Lines

☑ Stave lines

☑ Cell info. header

☑ Segments info. header

Figure 1. Extract loaded into the software

note to which it is unrelated in pitch instead of with the following note to which it relates. This aspect of Schenkerian theory has caused others (e.g. [9]), and myself in earlier work [5] to represent diminution (the opposite of reduction) as something which takes place in the intervals between notes rather than something applied to individual notes. This, however, results in graph structures which are not simple trees and so are much more difficult to process. (Of course, the connections which are absent in the tree structures are still present in the context dependencies, which bring their own complications, but this nevertheless appears to result in a more tractable structure.)

4. IMPLEMENTATION

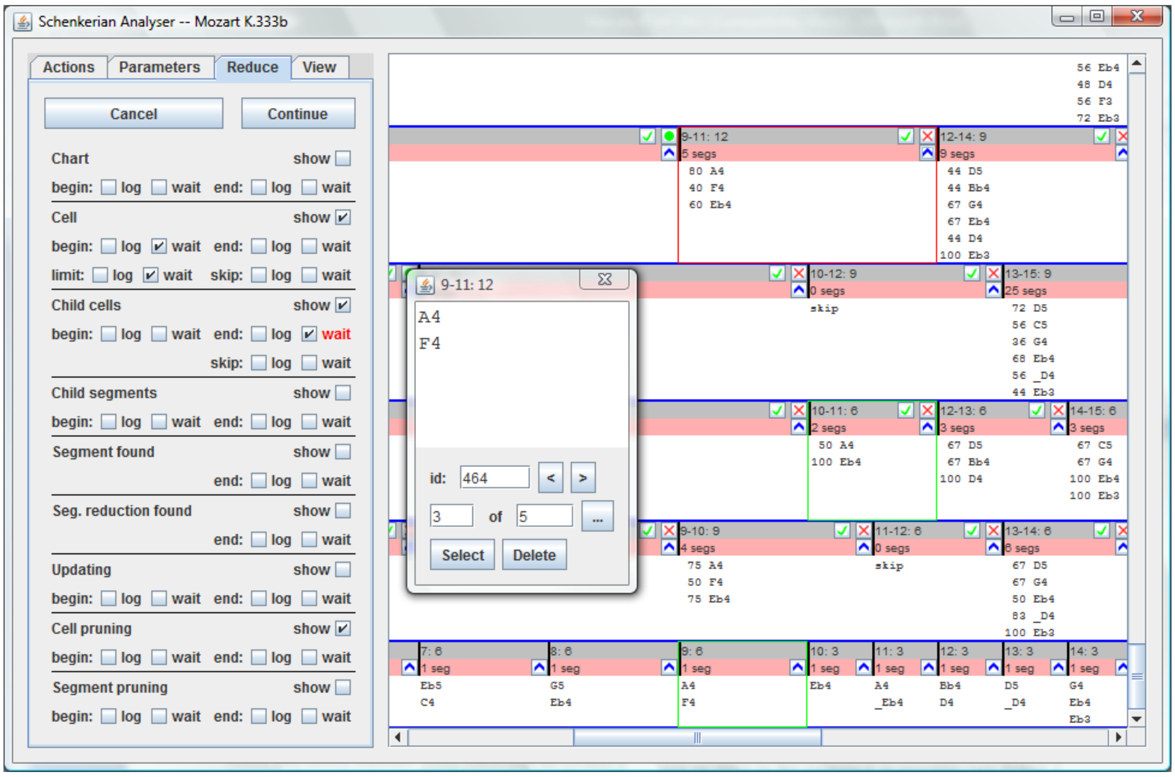
The fundamental problem in implementing Schenkerian analysis is computational complexity. As indicated in [6], deriving an analysis from a score is inherently of factorial complexity in time and space. At every stage of design of the software, therefore, computational efficiency has been emphasised. The general design is like a chart parser (a mechanism used in computational linguistics to reduce complexity in parsing [1]), which derives a 'chart' of multiple analyses in polynomial time and space. To extract a particular analysis from the chart is then a smaller process than to derive the analysis from scratch, but still one of exponential complexity.

The first step in analysis is to represent an extract of a score as a sequence of 'segments'. Each covers a distinct span of time, filling the interval between the preceding and following segments, and containing all the notes sounding in that interval. All notes in a segment last for the entire duration of the segment, and they can be tied to notes in the preceding and/or following segments. Thus long notes in the score are often split into several tied notes spread across a number of segments. A note is represented by its pitch alone. (Other characteristics such as dynamics and articulation are not irrelevant to Schenkerian analysis, but they appear to be of much lesser significance and have been ignored at present.)

The chart to be filled in the parsing stage is a triangular matrix whose bottom (longest) row of cells contains the segments of the surface of the piece. Each higher (and shorter) row will be filled with the segments which arise from reducing pairs of segments from the row(s) below, and have durations equal to the sum of the durations of 2, 3, 4 ... (according to the height of the row) of the segments of the surface below. The top row consists of a single cell which will eventually contain segments which span the entire extract. Cells in rows above the bottom can contain multiple segments, each representing alternative ways of reducing the segments below.

As the chart is filled and new segments are derived by reduction of pairs of 'child' segments, the links between parent and children, and their constraints, are recorded. This makes it possible to extract a complete analysis tree by following parent-child relations from a top-level segment, and also to ensure that a chart remains consistent when segments are deleted from it. Other information recorded with derived segments during parsing includes a putative 'goodness score' for the segment to facilitate selecting a 'best' analysis, and information about potential membership of an Ursatz.

Once the chart is filled, a number of analyses can be derived from it by selecting a segment in the top-level


Schenkerian Analyser -- Mozart K.333b

Actions | Parameters | Reduce | View

Cancel | Continue

Chart show ☐
begin: ☐ log ☐ wait end: ☐ log ☐ wait
Cell show ☑
begin: ☐ log ☑ wait end: ☐ log ☐ wait
limit: ☐ log ☑ wait skip: ☐ log ☐ wait
Child cells show ☑
begin: ☐ log ☐ wait end: ☐ log ☑ wait
skip: ☐ log ☐ wait
Child segments show ☐
begin: ☐ log ☐ wait end: ☐ log ☐ wait
Segment found show ☐
end: ☐ log ☐ wait
Seg. reduction found show ☐
end: ☐ log ☐ wait
Updating show ☐
begin: ☐ log ☐ wait end: ☐ log ☐ wait
Cell pruning show ☑
begin: ☐ log ☐ wait end: ☐ log ☐ wait
Segment pruning show ☐
begin: ☐ log ☐ wait end: ☐ log ☐ wait

```
56 Eb4
48 D4
56 F3
72 Eb3

9-11: 12    5 segs    80 A4  40 F4  60 Eb4
12-14: 9    9 segs    44 D5  44 Bb4  67 G4  67 Eb4  44 D4  100 Eb3

10-12: 9    0 segs    skip
13-15: 9    25 segs   72 D5  56 C5  36 G4  68 Eb4  56 _D4  44 Eb3

10-11: 6    2 segs    50 A4  100 Eb4
12-13: 6    3 segs    67 D5  67 Bb4  100 D4
14-15: 6    3 segs    67 C5  67 G4  100 Eb4  100 Eb3

9-10: 9     4 segs    75 A4  50 F4  75 Eb4
11-12: 6    0 segs    skip
13-14: 6    6 segs    67 D5  67 G4  50 Eb4  83 _D4  100 Eb3

7: 6   1 seg   Eb5 C4
8: 6   1 seg   G5 Eb4
9: 6   1 seg   A4 F4
10: 3  1 seg   Eb4
11: 3  1 seg   A4 _Eb4
12: 3  1 seg   Bb4 D4
13: 3  1 seg   D5 _D4
14: 3  1 seg   G4 Eb4 Eb3
```

9-11: 12
A4
F4
id: 464 < >
3 of 5 ...
Select | Delete

Figure 2. Part-way through reduction, showing text display and segment-information dialog

cell, then recursively selecting children until the surface is reached. Because of the context dependencies, naive selection does not always result in a valid analysis. Dependencies are therefore tracked, and the user is informed when no valid reduction remains. The dependencies also mean that the putative best score of a segment cannot always be realised.

5. USAGE

The software is written as an application in Java (version 1.6). The general principle of the user interface (see Figure 1) is a large area to display a visualisation of the emerging reduction on the right, and a set of tabbed panes on the left with controls for the display and for the reduction. The remainder of this paper illustrates use of the software to make a reduction of a short phrase from the last movement of Mozart piano sonata in B flat major, K.333.

The first step is to load an extract of music to be analysed. The software currently reads files which give information about the pitch and duration of notes in a simple text format. It is currently being adapted to read this information from MusicXML files. Reading from MEI and MIDI are planned for the future. Loading a file creates a new reduction chart with the bottom row filled with the notes of the extract and all other rows empty. This is the state of the software shown in Figure 1, where the display uses a format which shows horizontal bars on a stave to indicate the presence of those pitches. The vertical boundaries between cells are shown in the grey and pink headers. The grey header indicates which cell is shown by the corresponding start and end columns at the surface level and, following the colon, the duration of the cell as a multiple of the shortest duration found in the extract. Buttons in this header allow the entire contents of a cell to be rejected, or allow it to be selected, causing all overlapping cells to be rejected. The pink header shows the number of segments contained in each cell, and a button which brings up detailed information about the cells and their derivation. Cells showing an oblique line will be skipped in the course of reduction because they do not fall within the limits set on the ‘Parameters’ pane for syncopation or limits on the ratio of the durations of child segments.

One pane of controls allows the user to set what will be shown in the display during the reduction process, and where the software will pause to allow the user to interact. Figure 2 shows this pane and the state of reduction at a point where the cell covering columns 9-11 is being filled by deriving reductions from the surface segment in column 9 and the derived segments in the cell covering columns 10-11. The ‘parent’ cell is outlined in red and the two ‘child’ cells outlined in green. The overlaying dialog shows information about the third of the five segments so far derived for cell 9-11. Buttons allow the user to delete this segment, or to select it, deleting all others in the cell. The ‘...’ button brings up tables of other information about the harmonic constraints on the segment, its derivation and its score.

As mentioned above, cells in the reduction chart above the surface can contain a number of segments, representing different ways in which the music at the surface can be reduced. When segments are displayed in text, as in Figure 2, the percentage of segments in a cell which contain a particular pitch is indicated before the pitch. Thus 80% of the segments in cell 9-11 (four of the

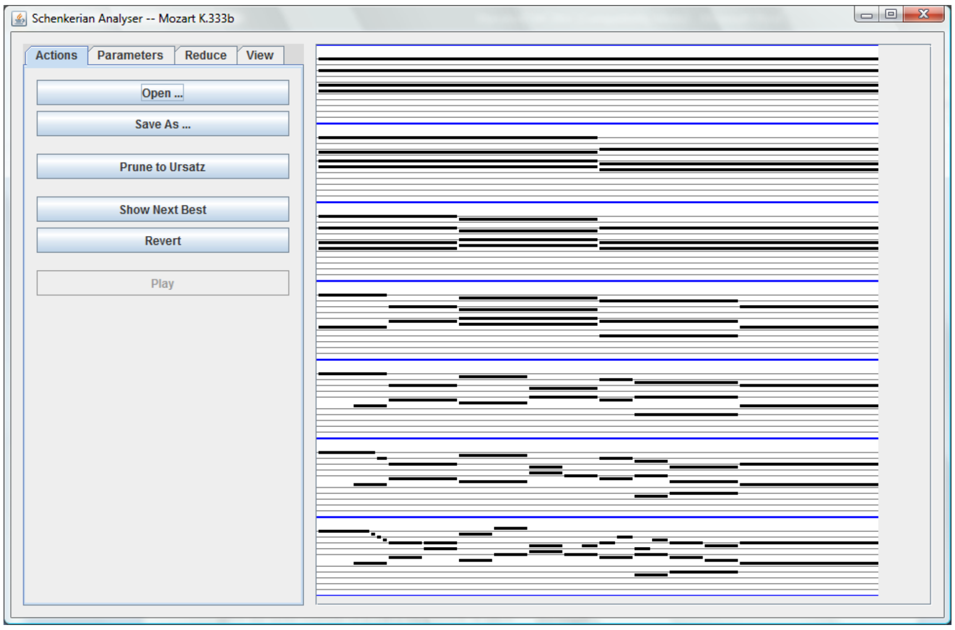


Figure 3. Completed analysis

five) contain the pitch A4, 40% F4 and 60% Eb4. In cases of a piano-roll like display (Figures 1 & 3), the darkness of a horizontal bar is related to the percentage of segments containing the corresponding pitch.

Once the entire chart is filled, the user can choose to have it pruned so that only segments which can participate in a complete Ursatz remain. Finally, a single 'best' analysis can be selected by clicking the 'Show Best' button. This launches a best-first search through the completed reduction chart for the highest-scoring tree of segments. Figure 3 shows the result of this. The headers have been removed from the display, and the sizes adjusted to allow the entire analysis to be displayed within the window. The 'Show Best' button has become 'Next Best'. Clicking this would replace the analysis shown with the next-highest-scoring analysis. The 'Revert' button causes the entire chart to be displayed once more.

The analysis shown in Figure 3 is not perfect, but it does conform quite well to the published analyses for this extract. The software does not perform so well for every extract, however. It is hoped that experimentation with this software, especially on extracts for which there exist previously published analyses, will allow development towards more reliably accurate analyses.

6. CONCLUSION

Development of the software continues. The latest version will be available for download at the author's web page (currently http://www.lancs.ac.uk/staff/marsdena).

The original primary aim of this project was theoretical—to discover the degree to which Schenkerian theory could be expressed in computational form—but a secondary aim has always been to facilitate software which behaves in a more intelligently musical way. Achievement of this is some way off still, and will have to await analysis software which is both more reliable and faster. The principal achievement of this version of the software is to make visible the reduction process which in earlier versions was entirely opaque.

7. REFERENCES

[1] Jurafsky, D., & Martin, J.H. *Speech and Natural Language Processing*, 2nd edition. Upper Saddle River, NJ: Pearson, 2009.

[2] Kassler, M. *A Trinity of Essays*. PhD dissertation, Princeton University, 1967.

[3] Kirlin, P.B. Using harmonic and melodic analyses to automate the initial stages of Schenkerian analysis. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, 2009, 423–428.

[4] Lerdahl, F., & Jackendoff, R. *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press, 1983

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

[6] Marsden, A. Automatic Derivation of Musical Structure: A Tool for Research on Schenkerian Analysis. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, 2007, 55–58.

[7] Marsden, A. Schenkerian Analysis by Computer: A Proof of Concept. *Journal of New Music Research*, *39*, 2010, 269–289.

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

[9] Yust, J.D. Formal Models of Prolongation. PhD dissertation, University of Washington, 2006.