LANCASTER INSTITUTE FOR THE CONTEMPORARY ARTS

Goldsmiths UNIVERSITY OF LONDON

Schenkerian Analysis as Search

Alan Marsden, Lancaster University

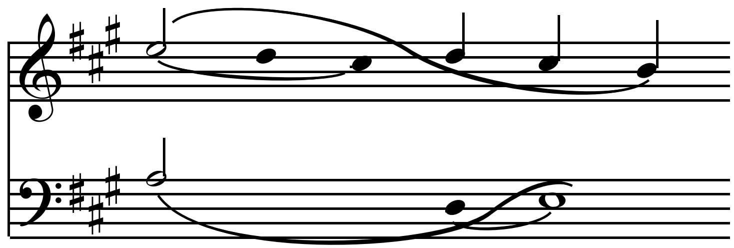
Geraint A. Wiggins, Goldsmiths, University of London

Schenkerian Analysis

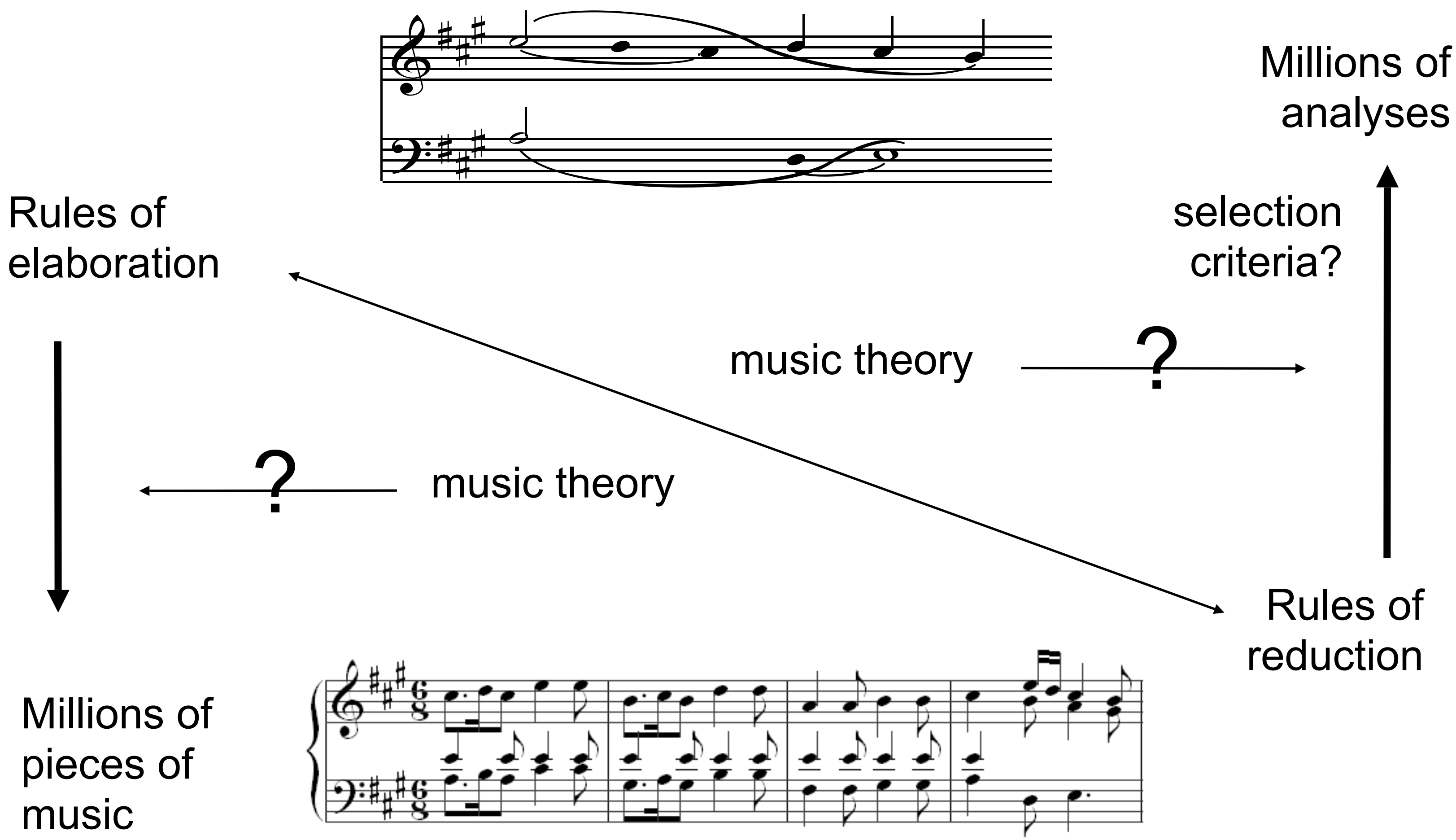
Progressively reduces a score, removing less essential features, to reveal the 'background' structure.

Mozart:

Schenker:



The Research Problem



Previous Work

- Kassler (1967, 1975, 1977, 1988)
  - program which successfully analyses three-voice middlegrounds
- Smoliar et al. (1976, 1978, 1980)
  - program capable of verifying an analysis
- Mavromatis & Brown (2004)
  - demonstration of theoretical possibility of Schenkerian analysis by context-free grammar
- Hamanaka, Hirata & Tojo (2005-7)
  - implementation of Lerdahl & Jackendoff reduction with adjustment of parameters (now moving towards automatic parameter-setting)
- Gilbert & Conklin (2007)
  - probabilistic grammar for melodic reduction

Formalisation of Reduction

- Marsden, (2005). 'Generative Structural Representation of Tonal Music', Journal of New Music Research, 34, 409–428
- All elaborations are binary:
  - elaborations producing more than one new note accommodated by special intermediate 'notes'
  - analysis is a set of binary trees, each corresponding roughly to a voice of the structure
  - trees can share nodes (one note can be elaborated in more than one way; a note can arise from more than one elaboration)

Formalisation of Reduction

- Elaborations generate new notes within the same time-span (cf. Lerdahl & Jackendoff, Komar).
- Only certain kinds of elaborations are possible.
- Elaborations have harmonic constraints.
- Some elaborations require specific preceding or following context notes.

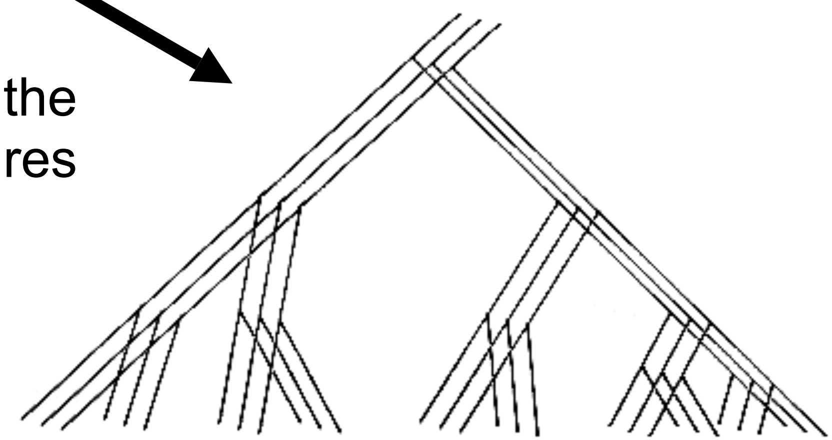
Basic Reduction Step

- For any pair of notes, given knowledge of the preceding notes (on the surface) and the following notes (both on the surface and at higher levels), we can determine:
  - which elaborations, if any, can produce these notes,
  - the parent note must be for each elaboration,
  - the requirements of key and harmony are for each elaboration.
- Given any pair of consecutive chords, information about preceding and following chords, and rules of harmonic and tonal consistency, we can determine the possible parent chords of that pair.

The Process

From the score …

… to derive the
tree structures

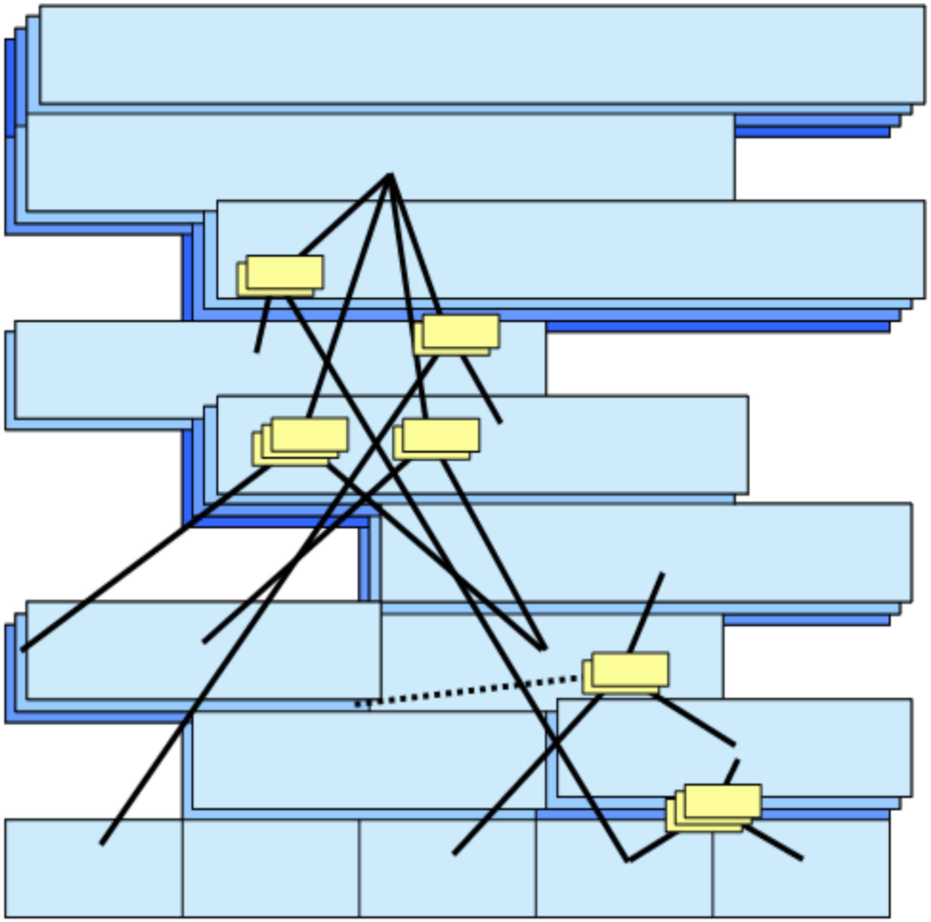


‘Chart-Parser’ Solution (CYK Algorithm)

- Similar to dynamic programming
- Construct a 3D matrix of valid local solutions.
  - lowest level is all the ‘chords’ of the surface of the piece:
    1D, n cells
  - higher levels are all possible chords derived by reduction from all possible pairs of chords below:
    2D, $(n - l) * x$ cells
    (l = level of reduction, x = unknown but limited number of possible local solutions)
- Any valid reduction tree can be derived from the matrix by selecting a top-level cell and then iteratively selecting pairs of possible children.

Solution Matrix

- A 'matrix' of local solutions, from which all possible reductions may be derived
- Complexity related to n^3



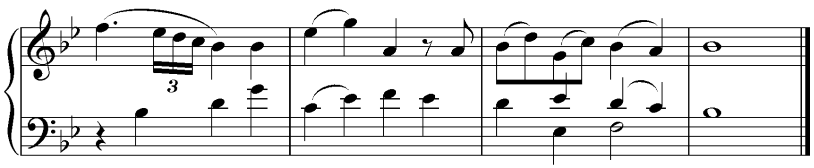
Example of Reduction Matrix

```
Row 5
0-5 16
 67 E5
 67 C5
 75 C4
 50 A3
 25 G3
Row 4
0-4 8        1-5 14
 63 E5        67 _E5
 38 D5        67 C5
 25 C4        75 C4
 50 B3        50 A3
 25 A3        25 G3
 38 G3
Row 3
0-3 7        1-4 6        2-5 12
 67 E5        33 _E5      100 C5
 33 D5        33 D5        75 C4
 33 C4        67 B3        50 A3
 33 B3        22 A3        25 G3
 50 A3        44 G3
Row 2
0-2 6        1-3 5        2-4 4        3-5 10
100 E5        50 _E5       43 D5       100 C5
 50 C4        30 D5        57 B3       100 C4
 25 B3        40 pB3-G3    14 A3        50 G3
 50 A3        40 B3        57 G3
              40 A3
Row 1
0-1 4        1-2 4        2-3 3        3-4 2        4-5 9
100 E5        67 _E5       50 D5       100 D5       100 C5
 33 pC4-A3    50 pB3-G3    50 B3        67 B3       100 C4
 33 C4        17 B3        50 A3        67 G3        50 G3
 33 B3        67 A3
Row 0
0 2          1 2          2 2          3 1          4 1          5 8
100 E5       100 _E5      100 A3       100 D5       100 _D5      100 C5
100 C4       100 B3                    100 B3       100 G3       100 C4
```

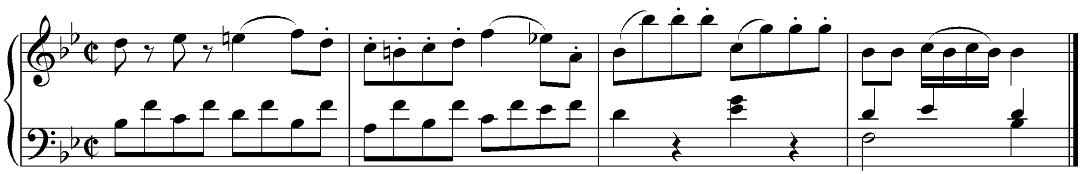
Problematic Size of Solution Space

Rondo themes from Mozart piano sonatas

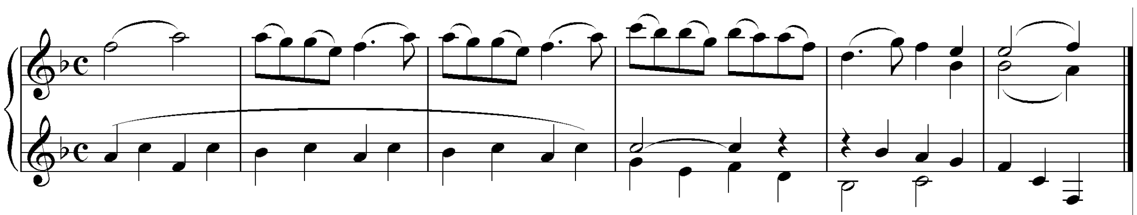
$5 * 10^{8}$ solutions, not including the ‘correct’ one



$7 * 10^{10}$ solutions, including the ‘correct’ one



$2 * 10^{20}$ solutions, including the ‘correct’ one



$7 * 10^{23}$ solutions, including the ‘correct’ one

Characterising the problem

- The problem is one of **combinatorial explosion**:
  - given a musical segment, many possible reductions can apply at any time
  - the order in which elaborations apply is often indeterminate (so there are many identical solutions under re-ordering)
  - many "valid" sequences of elaborations lead to non-sensical analyses
  - one does not know the solution in advance (not like, say, tic-tac-toe, where a winning board can be easily spotted}
- Because of this, an exhaustive computation method is never going to work in general
- Research question:
  - how far can we get using techniques from "Good Old-Fashioned Artificial Intelligence" (GOFAI)?

GOFAI search

- Early AI approach to problem solving
  - *symbolic representation* of states of the world in which a problem exists
  - *expansion* of (non-solution) states to give new states
  - *search algorithms* to explore routes between states
  - *solution detector* to identify success
- Like reading a road map in the dark with a small torch

GOFAI Search Implementation

- Four basic algorithms
  - Depth First (go all the way to the end of each path before trying the next)
  - Breadth First (go just one step along each path, iteratively, before trying the next)
  - Best First (evaluate each state and choose the best one each time, but keep all of them, and backtrack in the event of failure)
  - Algorithm A (estimate cost to solution and add it to cost of current state at each step, then search smallest first)
    - Algorithm A* (prove that estimate is *admissible*, so it is always less than actual cost, => guaranteed optimal search)
- Can all be implemented within one standard framework (see paper)

Schenkerian Reduction as A*/BFS search

- Formulation
  - Representation = Segmented score annotated with reduction
  - Transition = Schenkerian reduction on one pair of segments
  - Start state = Segmented initial score
  - End state = Well-formed Ursatz
- Heuristics
  - A* heuristic = number of states from start + minimal edit distance from current state to a well-formed Ursatz
    - measures progress in search
  - BFS heuristics from Marsden (various)
    - measure quality of current (partial) solution
  - BFS heuristics are used to choose between states with same A* heuristic

Implementation & Preliminary results

- Implemented "naively" in Prolog
  - not particularly fast
  - not able to cope with very large starting scores
  - very easy to understand the search and see what's going on
- Heuristics tested independently
  - A* heuristic does seem reliably to lead towards ursätze
  - Marsden's heuristics do seem to lead to good solutions
  - Together they seem to lead to good ursätze
- However,
  - works on small examples (so far)
  - needs further exploration

Preliminary results

- Comparison between heuristic and non-heuristic methods

Method	# nodes expanded to find **first** solution	# nodes expanded to find **best** solution
DepthFS	14	59
BreadthFS	36	60
A	14	16
A + BestFS	10	14

Further Work

- Matrix method
  - finding candidate heuristics from more and longer examples
- Search Method
  - prove A heuristic admissible (=> search optimal)
  - implement more realistically and test on larger scores
  - extend BestFS heuristics to include Marsden’s more recent work
- Combination method
  - combine them!

Supported by

Arts and Humanities Research Council (AHRC) research-leave award: ‘Analysing Musical Structure: Harmonic-Contrapuntal Reduction by Computer’ (AM)

Andrew W. Mellon Foundation project: ‘MeTAMuSE: Methodologies and Technologies for Advanced Musical Score Encoding’ (GW)