Proceedings of the 8th International Conference on Music Perception & Cognition, Evanston, IL, 2004
ICMPC8
S.D. Lipscomb, R. Ashley, R.O. Gjerdingen, & P. Webster (Eds.)
Adelaide, Australia: Causal Productions

PARSING CONTEXT-FREE GRAMMARS FOR MUSIC: A COMPUTATIONAL MODEL OF SCHENKERIAN ANALYSIS

Panayotis Mavromatis
New York University

Matthew Brown
Eastman School of Music

ABSTRACT

We outline a formalization and computer implementation of Schenker's theory of tonality. The theory of *Context-Free Grammars* and their parsing, which has been developed extensively by computer scientists and computational linguists, offers a natural framework to address our problem. We show how Schenker's prototypes and transformations can be cast in the form of a Context-Free Grammar. This allows us to implement for the first time a parsing algorithm that automates the analytical process. We develop a computer program using Prolog's *Definite Clause Grammar* formalism. The program parses a piece presented in a suitable encoded form, producing a tree that represents a Schenkerian analysis of that piece. If more than one syntactically correct parsing is possible, the program produces corresponding analyses in separate trees. We discuss our work's implications for music theory and music psychology.

BACKGROUND AND AIMS

For over twenty five years, music theorists have been trying to implement Schenkerian theory and analysis on the computer; Kassler,[1] Smoliar,[2] Snell,[3] and others, have written programs based on Schenkerian principles. And with good reason. By explaining tonal relations in terms of prototypes and transformations, Schenker's work is particularly well suited to implementation on the machine. In return, implementation on the computer allows music theorists to check the consistency and completeness of the theory, as well as to track the enormous number of transformations required to generate complex tonal surfaces.

Remarkable though they may be, the programs devised by Kassler, Smoliar, and Snell leave many complex but important issues open. For one thing, they either model portions of Schenker's original theory, or they adapt it in significant ways. For another, although they offer the *data structures* required to represent the theory's prototypes and transformations, they do not provide *algorithms* that implement the analytical process; the parsing of a tonal piece to produce a Schenkerian derivation is left to the user.

Given the formal similarity of Schenker's system to phrase structure grammar,[4] it is perhaps surprising that Kassler, Smoliar, and Snell did not take advantage of existing formalisms developed in the fields of computational linguistics and the theory of formal languages. This would have given them access to ready-made tools for addressing the complex issues posed by the theory in a way that an ad hoc computational framework cannot.

Advances in Schenkerian theory,[5,6,7] and the music theory community's expanding interest in more sophisticated computational techniques, have invited us to revisit the problem. Our goal has been to cast Schenkerian theory in the form of a *Context-Free Grammar* (CFG). The theory of CFG's and their parsing has been developed extensively by computer scientists and computational linguists.[8,9,10] If we can produce an exact correspondence between Schenkerian transformations and the rewrite rules of a CFG, a number of computational frameworks become immediately available to us for implementing parsing algorithms that automate the analytical process. These frameworks include *Augmented Transition Networks*, *Definite Clause Grammars*, and *Chart Parsers*.[9,10]

MAIN CONTRIBUTION

Perhaps the most important attempt to make an explicit connection between Schenker's transformations and context-free rewrite rules was offered by Keiler.[11,12] A more careful examination, however, reveals that Keiler's simple scheme runs into a number of problems:

- **Problem 1** Some Schenkerian transformations (e.g., *deletion* and *displacement*) do not involve rewriting a simple entity into many, and hence cannot be represented by context-free rewrite rules.
- **Problem 2** Those transformations that *can* (e.g., *arpeggiation* and *linear progression*) aren't manifestly context-free—their applicability is often limited by context, e.g., when parallel perfect fifths/octaves result from their application.

- **Problem 3** Since context-free grammars were originally conceived for languages that generate one-dimensional strings, they cannot handle polyphonic configurations without some serious adjustment.

By addressing all three problems, we are able to demonstrate for the first time that Schenkerian theory is a context-free grammar. The following paragraph outlines our argument. Details will be provided in a future publication.

Putting aside for the moment Problem 1, we can avoid Problem 2 by moving from a *note-based* representation to an *interval-based* one. Individual Schenkerian transformations can then be combined into consonant polyphonic combinations that simultaneously apply to the same time span. For instance, a chord can be expanded into a combination of linear and neighbor note patterns, each of them applying to the individual members of the chord. It is such polyphonic clusters of transformations, rather than the individual transformations, that correspond to rewrite rules. In Schenkerian theory, one member of the cluster is singled out as primary (e.g., a *primary span*), and can be used to identify/label the rewrite rule; the remaining members of the cluster provide consonant harmonization to the primary transformation. In this light, Problem 1 can be solved by allowing transformations such as *delete* and *displacement* to be part of clusters; even though they can never be the primary transformations of the cluster, *delete* and *displacement* can act on the notes generated by the other members of the polyphonic combination, giving rise to modified rewrite rules.

As the implementation framework, we chose Prolog's *Definite Clause Grammar* (DCG) formalism. Prolog is a high level programming language that supports the declarative programming paradigm: a program is a direct representation of the knowledge needed to solve a problem in terms of objects and relations. Procedures need not be explicitly coded, and the execution of the program is driven by Prolog's built in engine. DCG is a syntactic extension of Prolog that allows one to express directly a CFG as a set of rewrite rules. Parsing DCG's is built into the Prolog system, so the programmer need not specify the algorithm explicitly. This allowed us to focus on the formal theoretical issues involved without worrying about the implementation details. The program consisting of the DCG rules can parse a piece presented in a suitable encoded form, producing a tree that represents a Schenkerian analysis of that piece. If more than one syntactically correct parsing is possible, the program produces corresponding analyses in separate trees.

IMPLICATIONS

We believe that our approach has led to a better understanding of theoretical issues in Schenkerian theory, most notably the relations among simultaneous transformations, the *harmonize* operation, and the status of *deletion* and *displacement*. By producing a working computer system that performs analysis, we are introducing an important tool in the music theorist's toolbox, a tool that provides a formal framework for formulating and testing extensions to Schenker's theory, e.g., in the domain of rhythm, or in the realm of early and late tonality. A computer tool may also make possible the analysis of large corpora in search of, e.g., stylistic traits. Last, but not least, a formal model of Schenkerian theory allows us to address psychological issues, most notably the question of whether it can shed light on tonal compositional expertise.

TOPIC AREAS

Computational Models
Schenkerian Theory
Music Analysis

REFERENCES

1. Kassler, M. "Explication of the Middleground of Schenker's Theory of Tonality." *Miscellanea Musicologica* 9 (1977): 72-81.
2. Smoliar, S. "A Computer Aid for Schenkerian Analysis." *Computer Music Journal* 4 (1980): 41-59.
3. Snell, J. "Design for a Formal System for Deriving Tonal Music." Master's Thesis, SUNY Binghamton, 1979.
4. Chomsky, N. *Aspects of the Theory of Syntax.* Cambridge, MA: M. I. T. Press, 1965.
5. Brown, M. "A Rational Reconstruction of Schenkerian Theory." Ph.D. Thesis, Cornell University, 1989.
6. Brown, M. "Rothstein's Paradox and Neumeyer's Fallacies." *Intégral* 12 (1998): 95-132.
7. Brown, M. *Explaining Tonality: A Schenkerian Perspective.* Rochester, NY: University of Rochester Press, forthcoming.
8. Hopcroft, J. and Ullman, J. D. *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1979.
9. Gazdar, G. and Mellish, C. *Natural Language Processing in Prolog: An Introduction to Computational Linguistics.* Reading, MA: Addison-Wesley, 1989.
10. Allen, J. *Natural Language Understanding.* 2nd ed. Redwood City, CA: Benjamin Cummings, 1995.
11. Keiler, A. "The Syntax of Prolongation (Part I)." *In Theory Only* 3/5 (1976): 3-27.
12. Keiler, A. "On Some Properties of Schenker's Pitch Derivations." *Music Perception* 1/2 (1984): 200-228.