Automatic Functional Harmonic Analysis

W. Bas De Haas,* José Pedro Magalhães,† Frans Wiering,* and Remco C. Veltkamp*

*Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht, The Netherlands
{W.B.deHaas, F.Wiering, R.C.Veltkamp}@uu.nl
†Department of Computer Science
University of Oxford
Wolfson Building, Parks Road
Oxford OX1 3QD, United Kingdom
jpm@cs.ox.ac.uk

Abstract: Music scholars have been studying tonal harmony intensively for centuries, yielding numerous theories and models. Unfortunately, a large number of these theories are formulated in a rather informal fashion and lack mathematical precision. In this article we present HarmTrace, a functional model of Western tonal harmony that builds on well-known theories of tonal harmony. In contrast to other approaches that remain purely theoretical, we present an implemented system that is evaluated empirically. Given a sequence of symbolic chord labels, HarmTrace automatically derives the harmonic relations between chords. For this, we use advanced functional programming techniques that are uniquely available in the Haskell programming language. We show that our system is fast, easy to modify and maintain, robust against noisy data, and that its harmonic analyses comply with Western tonal harmony theory.

For ages, musicians, composers, and musicologists have proposed theories regarding the structure of music to better understand how it is perceived, performed, and appreciated. In particular, tonal harmony exhibits a considerable amount of structure and regularity. The first theories describing tonal harmony date back at least to the 18th century (Rameau 1722). Since then, a rich body of literature that aims at explaining the harmonic regularities in both informal and formal models has emerged (e.g., Lerdahl and Jackendoff 1996). Such models have attracted numerous computer music researchers to investigate the automation of the analysis and generation of harmony. Most of these theories, however, have proven to be very hard to implement (e.g., Clarke 1986). We are not aware of a model that has a working implementation that effectively analyzes tonal harmony and deals robustly with noisy data, while remaining simple and easy to maintain, and scaling well to handle musical corpora of considerable size. In this article we present HarmTrace (Harmony Analysis and Retrieval of Music with Type-level Representations of Abstract Chord Entities), a system that meets these requirements using state-of-the-art functional programming techniques. HarmTrace allows us to easily adapt the harmonic specifications, empirically evaluate the harmonic analyses, and use these analyses for tasks such as similarity estimation and automatic annotation of large corpora.

The HarmTrace harmony model draws on the ideas of Rohrmeier (2007, 2011). Rohrmeier modeled the core rules of Western tonal harmony as a (large) context-free grammar (CFG, see Chomsky 1957). Later, De Haas et al. (2009) implemented this grammar and specifically tuned it for jazz harmony, with the aim of modeling harmonic similarity. The HarmTrace system transfers these ideas to a functional setting, solving typical problems that occur in context-free parsing (e.g., the rejection of pieces that cannot be parsed) and controlling the number of ambiguous solutions. Because it relies on advanced functional programming techniques not readily available in most programming languages, HarmTrace is inextricably bound to Haskell (Peyton Jones 2003). Haskell is a purely functional programming language with strong static typing. It is purely functional because its functions, like regular mathematical functions, are guaranteed to

Computer Music Journal, 37:4, pp. 37–53, Winter 2014
doi:10.1162/COMJ_a_00209

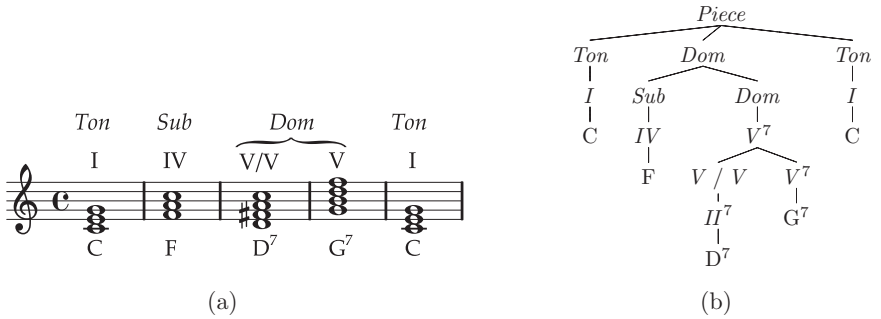


Figure 1. A typical chord sequence (a) and its harmonic analysis, as generated by HarmTrace (b). The chord labels are printed below the score, and the scale degrees and functional analysis above the score. For simplicity, we ignored voice-leading. Ton, Dom, *and* Sub *denote tonic, dominant, and subdominant, respectively.*

produce the same output when given the same input. It is strongly typed because it enforces restrictions on the arguments to functions, and it does so statically (i.e., at compilation time). Through its main implementation, the Glasgow Haskell Compiler (GHC, http://haskell.org/ghc), Haskell offers state-of-the-art functional programming techniques, like error-correcting combinator parsers, type-level computations, and *datatype-genericity* (polytypic functions), that are not available in any other mainstream language. These features proved to be essential to HarmTrace, as we will show.

Naturally, any Turing-complete language can be used to implement a particular algorithm or model; this also holds for implementing HarmTrace. Most programming languages, however, do not offer features, such as type-indexed computations, that make implementing HarmTrace much easier. We prefer Haskell over, e.g., C, Java, or LISP, because its strong type system gives us better guarantees of correctness, makes the development process clearer and easier, and allows for compiler type-directed optimizations (e.g., fusion laws). Other non-mainstream programming languages with type computations exist, such as Coq and Agda, but they do not have practical library support.

Following Rohrmeier, a core assumption that underlies our harmony model is that Western tonal harmony is organized hierarchically and transcends Piston's table of usual root progressions (Piston 1991, ch. 3, p. 21). As a consequence, within a sequence of chords some chords can be removed because of their subordinate role, leaving the global harmony structure intact, whereas removing other chords can significantly change how the chord sequence is perceived. This is illustrated in the sequence displayed in Figure 1a: the D^7 chord in this sequence can be removed without changing the general structure of the harmony, although removing the G^7 or the C at the end would cause the sequence to be perceived very differently. This implies that within a sequence not all chords are equally important, and must be organized hierarchically. This hierarchical organization is reflected in the tree structure shown in Figure 1b. The subdominant F has a subordinate role to the dominant G^7, which is locally prepared by a secondary dominant D^7. The tonic C releases the harmonic tension built up by the F, D^7, and G^7.

As in our earlier work with Rohrmeier (De Haas et al. 2009), the development of HarmTrace has been driven by its application in content-based music-information retrieval (MIR, Downie 2003) research. Within MIR, the notion of musical similarity plays a prominent role because it allows ordering musical pieces in a corpus. Using such an ordering, one can retrieve harmonically related pieces, like cover songs, classical variations, or all blues pieces in a corpus. For performing such searches, a measure of harmonic similarity is essential. Because our system captures the global and local relations between chords, it can be used to improve systems that can benefit from this contextual information. It has been shown that analyzing the hierarchical relations between chords in a sequence improves the quality of a harmonic-similarity measure in retrieval tasks (De Haas et al. 2009, 2011; Magalhães and De Haas 2011). Another application area for HarmTrace is automatic chord transcription (De Haas, Magalhães, and Wiering 2012). This is a common MIR task that aims at recognizing a chord sequence from the

audio signal. For every beat position, the chords that match the spectrum well are selected from a chord dictionary. Subsequently, the HarmTrace model is used to select the candidate sequence that best fits the modeled rules of tonal harmony. Chordify, an Internet startup, uses HarmTrace to make automatic chord transcription accessible to the general public (http://chordify.net, see also De Haas et al. 2012).

The application to MIR explains some of the choices made in the development of HarmTrace. In particular, because a large corpus of chord sequences, mainly from the jazz repertoire, is available for retrieval tasks, the harmony model exhibits a bias towards jazz harmony. In this article we describe the musical aspects of HarmTrace; its Haskell-specific implementation aspects are described elsewhere (Magalhães and De Haas 2011).

A fully functional model of tonal harmony that can quickly analyze chord sequences offers several other benefits. Musicologists study the harmonic structure of pieces and annotate them by hand. This is a time-consuming enterprise, especially when large corpora are involved. With the automatic annotation techniques that we present here, this can be done quickly, even for large corpora possibly containing errors. Additionally, HarmTrace could aid in (automatic) composition by generating sequences of chords, generating harmonically realistic continuations given a sequence of chords, or automatically harmonizing a melody (Koops, Magalhães, and De Haas 2013).

This article starts by discussing a relevant selection of the large body of existing literature on harmony theory and modeling in the next section. Subsequently, we explain our harmony model, followed by an evaluation of some detailed, example analyses created by this model. Next, we show that HarmTrace can deal with large amounts of noisy data. Finally, we conclude the article by discussing the limitations of our system and differences from other models.

Related Work

The 19th and 20th centuries have yielded a wealth of theoretical models of Western tonal music; tonal harmony, in particular, has received considerable attention. Most theories that describe the relationships between sequential chords capture notions of order and regularity; some combinations of chords sound natural whereas others sound awkward (e.g., Rameau 1722). These observations led music theorists to develop ways to analyze the function of a chord in its tonal context (e.g., Riemann ca. 1895). Unfortunately, the majority of these theories are formulated rather informally and lack descriptions with mathematical precision. In this section we give a condensed overview of the theories that played an important role in the formation of the harmony model we present in this article.

Lerdahl and Jackendoff's *Generative Theory of Tonal Music* (GTTM, 1996) is a seminal work that further formalized the ideas of Schenker (1935). GTTM structures Western tonal compositions by defining recursive, hierarchical dependency relationships between musical elements using well-formedness and constraint-based preference rules. The GTTM framework distinguishes four kinds of hierarchical structure: meter, grouping, time-span reduction, and prolongational reduction. Although GTTM can be considered one of the greatest contributions to music theory and music cognition of the last few decades, implementing the theory is difficult because the often vague and ambiguous preference rules lead to a wide range of possible analyses (Clarke 1986; Temperley 2001, ch. 1; Hamanaka, Hirata, and Tojo 2006).

The recursive formalization proposed by Lerdahl and Jackendoff suggests a strong connection between language and music. Many other authors have argued that tonal harmony should be organized in a hierarchical way similar to language, leading to numerous linguistically inspired models since the 1960s (Roads 1979). One of the pioneers to propose a grammatical approach to harmony was Winograd (1968). More recently, Steedman (1984, 1996) modeled the typical twelve-bar blues progression with a categorial grammar; Chemillier (2004) elaborates on these ideas by transferring them to a CFG. Similarly, Pachet (1999) proposes a set of rewrite rules for jazz harmony comparable to Steedman's grammar. Pachet furthermore shows that these rules can be learned from chord sequence data in an automated

fashion. Additionally, quasi-grammatical systems for Schenkerian analysis have been proposed recently (Marsden 2010). Furthermore, Choi (2011) developed a system for analyzing the harmony of jazz chord sequences; this system identifies common harmonic phenomena, like secondary dominants and tritone substitutions, and labels the chords involved accordingly. The relation between music and language is not merely a theoretical one; a growing body of evidence, also from neuropsychology and neuroimaging, suggests that music and language are more closely related than was previously believed (Patel 2003). An in-depth overview of the animated debate on the relation between music and language is beyond the scope of this article, however.

The generative formalism proposed by Rohrmeier (2007, 2011), which the HarmTrace model draws extensively from, expands upon many of these earlier approaches in a number of ways. Rohrmeier gives an encompassing account of how relationships in tonal harmony can be modeled using a generative CFG with variable binding. His grammar models form, phrasing, theoretical harmonic function (Riemann ca. 1895), prolongation of scale degrees (Schenker 1935; Lerdahl and Jackendoff 1996), and modulation. Rohrmeier's model differs from earlier grammatical formalisms in various ways. Steedman's approach (1984, 1996) uses seven context-sensitive rules (with variations) to explain the structure of blues progressions. Steedman's grammar does not go beyond the jazz blues domain, however, and it remains unclear how it could be adapted to support other styles. Rohrmeier's formalism also differs from GTTM: The latter aims at describing the core principles of tonal cognition, and harmony is covered mainly as a prolongational phenomenon, whereas Rohrmeier's formalism describes the structure of tonal harmony from a music-theoretical perspective with concrete context-free rules. Rohrmeier acknowledges that a full account of tonal harmony would require a large number of varying style-specific rules, and his formalism aims to capture only the core rules of Western tonal harmony.

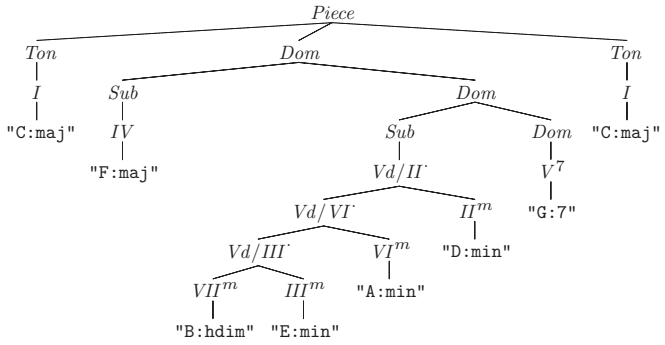
De Haas et al. (2009) performed a first attempt to implement the ideas of Rohrmeier. Although the results were promising, the context-free parsing techniques used in that work hampered both theoretical as well as practical improvements. First, a sequence of chords that did not precisely match the context-free specification was rejected and no explanatory information was given to the user. For example, appending one awkward chord to an otherwise grammatically correct sequence of chords forced the parser to reject the complete sequence, not returning any partial information about what it had parsed.

Second, musical harmony is ambiguous and chords can have multiple meanings depending on the tonal context in which they occur. This is reflected in all the grammatical models discussed earlier. A major drawback of CFGs is that they are very limited in ways of controlling the ambiguity of the specification. It is possible to use rule-weightings and to set low weights to rules that explain rare phenomena. This allows for ordering the ambiguous solutions by the total relevance of the rules used. This does not overcome the fact that, for some pieces, the number of parse trees grows exponentially as function of the number of input chords.

Finally, writing context-free rules by hand is a tedious and error-prone enterprise, especially because the grammatical models can become rather large. For instance, a rule generalizing over an arbitrary scale degree has to be expanded for each scale degree, I, II, III, etc. Hence, some form of high-level grammar generation system is needed to allow for generalizing over scale degree and chord type, and to control conditional rule execution.

Another important model that has influenced the development of HarmTrace is that of Temperley (2001) and Temperley and Sleator (1999). They give an elaborate formal account of Western tonal music, and also provide an efficient implementation. This rule-based system, which is partly inspired by GTTM, can perform an analysis of the chord roots and the key given a symbolic score, but does not formalize the hierarchical relations between chords. Our system continues where Temperley's left off: Based on the available chord and key information of the piece, we model the global and local dependency relations between these chords.

Figure 2. An example of a diatonic cycle-of-fifths progression in C major. The leaves represent the input chords, using the syntax of Harte et al. (2005), and the internal nodes denote the harmonic structure. The $Vd/\mathfrak{X}$ nodes represent diatonic fifth successions.



The HarmTrace System

In this section we explain how we model the regularities and hierarchical dependencies of tonal harmony. HarmTrace transfers the ideas developed by De Haas et al. (2009) to a setting based on functional programming. Whereas the contributions of the majority of models we discussed in the previous section are purely theoretical, we present a system that can be evaluated empirically and is usable in practice. This comes at a price, however: Our present model does not support full modulation, i.e., modulation to every possible key. The model can only handle change of mode—going from major to minor or vice versa—without changing the root of the key. (We also use the term *parallel keys* to describe this relationship.) As a consequence, this requires the model to have information about the key of the piece. Also, because we mainly use jazz-oriented input data in this article, we also include some specifications that describe typical phenomena in jazz harmony. Figure 2 shows an example analysis as produced by HarmTrace. The chord labels that were used as input are the leaves of the tree, and the internal nodes represent the harmonic relations between the chords.

Music, and harmony in particular, is intrinsically ambiguous; certain chords can have multiple meanings within a tonal context. Although a model of tonal harmony should reflect some ambiguity, defining many ambiguous specifications can make the number of possible analyses grow exponentially for certain chord progressions. In most of the ambiguous cases it is clear from the context which of the possible solutions is the preferred one, however. Hence, we can select the favored analysis by constraining the application of the specification leading to the undesired analysis. We are able to do so because we rely on strongly typed data structures, which allow us to formulate our rules in a very precise manner. In cases where it is less clear from the context which solution is preferred, we accept a small number of ambiguous analyses.

The HarmTrace system explores the relations between (generalized) algebraic data types and context-free production rules. A CFG defines a language: Given a set of production rules and a set of words (or *tokens*), it accepts only combinations of tokens that are valid sequences of the language. The notion of an algebraic datatype is central in Haskell. Similarly to a CFG, a datatype defines the structure of values that are accepted. Hence, a collection of datatypes can be viewed as a very powerful CFG: The type-checker accepts a combination of values if their structure matches the structure prescribed by the datatype, and rejects this combination if it does not. Within HarmTrace, the datatypes represent the relations between the structural elements in tonal harmony, and the chords are the values. An important difference between a CFG and a Haskell datatype, however, is that datatypes provide more much modeling freedom and control, especially the generalized algebraic datatypes (Schrijvers et al. 2009) that we use. These datatypes allow us to restrict the number of applications of a specification, constrain the conditions for application, and order the specifications by their importance. Furthermore, they allow for the definition of specifications that only apply to certain keys or modes, the exclusion of the application of transposition functions to specific scale degrees (e.g., Spec. 18, described later), and the preference of certain specifications over others. For technical details, we refer to Magalhães and De Haas (2011). The source code is available online at http://hackage.haskell.org/package/HarmTrace-0.6.

A Model of Tonal Harmony

We now elaborate on how our harmony datatypes are organized. Haskell knowledge is not required to

understand our model: We use a simplified syntax to describe the datatype specifications that is inspired by the syntax used to describe CFGs. Users of HarmTrace need only know Haskell if they wish to change the model in any way; simply using the current model to obtain analyses can be done without any Haskell knowledge. In the following subsections we group related datatype specifications (abbreviated as Spec.), and explain their musical interpretation and background. First, we introduce a variable $\mathfrak{M}$ for the mode of the key of the piece, which can be major or minor. The mode variable is used to parametrize all the specifications of our harmonic datatype specification; some specifications hold for both modes (inidicated by $\mathfrak{M}$), whereas other specifications hold only for the major or minor mode (respectively indicated by "Maj" and "Min"). Similarly to a CFG, we use a | to denote alternatives, and a + to represent optional repetitions of a datatype.

Functional Categories

1. $Piece_{\mathfrak{M}} \rightarrow Func^{+}_{\mathfrak{M}}$
2. $Func_{\mathfrak{M}} \rightarrow Ton_{\mathfrak{M}} \mid Dom_{\mathfrak{M}}$ $\quad \mathfrak{M} \in \{\text{Maj}, \text{Min}\}$
3. $Dom_{\mathfrak{M}} \rightarrow Sub_{\mathfrak{M}}\ Dom_{\mathfrak{M}}$

Spec. 1–3 define that a valid chord sequence, $Piece_{\mathfrak{M}}$, consists of at least one and possibly more functional categories. A functional category classifies chords as being part of a tonic ($Ton_{\mathfrak{M}}$), dominant ($Dom_{\mathfrak{M}}$), or subdominant ($Sub_{\mathfrak{M}}$) structure, where a subdominant structure must always precede a dominant structure. These functions constitute the top-level categories of the harmonic analysis and model the global development of tonal tension: A subdominant builds up tonal tension, the dominant exhibits maximum tension, and the tonic releases tension. The functional categories capture the idea proposed by Riemann (ca. 1895) that every chord within a tonality can be reduced to one of these harmonic functions. The order of the dominants and tonics is not constrained by the model, and they are not grouped into larger phrases.

From Functions to Scale Degrees

4. $Ton_{\text{Maj}} \rightarrow I_{\text{Maj}} \mid I_{\text{Maj}} IV_{\text{Maj}} I_{\text{Maj}} \mid III^{m}_{\text{Maj}}$
5. $Ton_{\text{Min}} \rightarrow I^{m}_{\text{Min}} \mid I^{m}_{\text{Min}} IV^{m}_{\text{Min}} I^{m}_{\text{Min}} \mid \flat III_{\text{Maj}}$
6. $Dom_{\mathfrak{M}} \rightarrow V^{7}_{\mathfrak{M}} \mid V_{\mathfrak{M}}$
7. $Dom_{\text{Maj}} \rightarrow VII^{m}_{\text{Maj}}$ $\quad \mathfrak{M} \in \{\text{Maj}, \text{Min}\}$
8. $Dom_{\text{Min}} \rightarrow \flat VII_{\text{Min}}$
9. $Sub_{\mathfrak{M}} \rightarrow II^{m}_{\mathfrak{M}}$
10. $Sub_{\text{Maj}} \rightarrow IV_{\text{Maj}} \mid III^{m}_{\text{Maj}} IV_{\text{Maj}}$
11. $Sub_{\text{Min}} \rightarrow IV^{m}_{\text{Min}} \mid \flat III_{\text{Min}} IV_{\text{Min}}$

Spec. 4–11 translate the tonic, dominant, and subdominant datatypes into scale-degree datatypes. Apart from a few minor differences, Spec. 4–11 follow Rohrmeier's (2007) model. A tonic translates to a first degree, a dominant to a fifth degree, and the subdominant to a fourth degree in both major and minor keys. We denote scale-degree datatypes with Roman numerals, but because our model jointly specifies datatypes for major as well as minor mode, we deviate from notation that is commonly used in classical harmony and represent scale degrees as intervals relative to the diatonic major scale. For example, III^{m} unequivocally denotes the minor chord built on the note a major third interval above the key's root and does not depend on the mode of the key. Similarly, a $\flat III$ denotes a major chord built on the root a minor third interval above the key's root.

A scale degree datatype is parametrized by a mode, a chord class, and the scale degree, i.e., the interval between the chord root and the key root. The chord class categorizes scale degrees as one of four types of chords (denoted in lowercase with superscripts) and is used to constrain the application of certain specifications, e.g., Spec. 16–17 (defined later). The four classes are major (no superscript), minor (m), dominant seventh (7), and diminished (0). Chords in the minor class contain a minor or diminished triad and can have possibly altered or non-altered additions, except for the diminished seventh. Chords categorized as major contain a major triad and can be extended by non-altered additions, with the exception of the dominant seventh chord (with additions). Chords

of the dominant class have a major triad and a minor seventh, or an augmented triad. Chords in the dominant class can be extended with altered or non-altered notes. Finally, the diminished class contains only the diminished seventh chord. The usage of chord classes allows us to define certain specifications that only hold for dominant chords, whereas other specifications might hold only for minor chords, etc.

Tonics can furthermore initiate a plagal cadence (Spec. 4–5). We deliberately chose to model the plagal cadence with scale degrees and not with *Sub* and *Ton* because this reduces the number of possible analyses. Also a $Sub_{\mathfrak{M}}$ can translate into a $II^{m}_{\mathfrak{M}}$ because of its preparatory role, and the dominant translates into the seventh scale degree, VII^{m}_{Maj} ($\flat VII_{\text{Min}}$ in minor). Similarly, we could have chosen to model the $Sub_{\mathfrak{M}}$ to translate also to VI^{m}_{Maj} ($\flat VI_{\text{Maj}}$ in minor). We chose to solve this by creating specifications for chains of diatonic fifths instead (Spec. 18–19; see, for instance, Figure 2).

The $Ton_{\mathfrak{M}}$ translating into III^{m}_{Maj} ($\flat III_{\text{Min}}$ in minor) is perhaps the most unexpected transformation (and we deviate from Rohrmeier's 2007 model here). Often the third degree can be explained as either a preparation of a secondary dominant (Spec. 17), as being part of diatonic chain of fifths (Spec. 19), or as supporting the subdominant (Spec. 10–11). In certain progressions, however, it cannot be explained by any of these specifications, and is best assigned a tonic function because it has two notes in common with the tonic and does not create strong harmonic tension.

From Scale Degrees to Chords

12. $I_{\text{Maj}} \rightarrow$ `"C:maj"` | `"C:maj6"` | `"C:maj7"` | `"C:maj9"` | ...
13. $I^{m}_{\text{Min}} \rightarrow$ `"C:min"` | `"C:min7"` | `"C:min9"` | `"C:min(13)"` | ...
14. $V^{7}_{\text{Maj}} \rightarrow$ `"G:7"` | `"G:7(b9,13)"` | `"G:(#11)"` | `"G:7(#9)"` | ...
15. $VII^{0}_{\mathfrak{M}} \rightarrow$ `"B:dim(bb7)"`

Finally, scale degrees are translated into the actual surface chords, which are used as input for the model. The chord notation used is that of Harte et al. (2005). The conversions are trivial and illustrated by a small number of specifications above, but the model accepts all chords in Harte's syntax. The model uses a key-relative representation; in Spec. 12–15 we used chords in the key of C. Hence, a I_{Maj} translates to the set of C chords with a major triad, optionally augmented with additional chord notes that do not make the chord minor or dominant. Similarly, V^{7}_{Maj} translates to all G chords with a major triad and a minor seventh, etc. To simplify the treatment of chords differing only by these additional chord notes, we cluster chords with the same class and same scale degree (e.g., `"C:min7" "C:min9"`) in one datatype.

Secondary Dominants

16. $\mathfrak{X}^{\mathfrak{C}}_{\mathfrak{M}} \rightarrow V/\mathfrak{X}^{7}_{\mathfrak{M}}\ \mathfrak{X}^{\mathfrak{C}}_{\mathfrak{M}} \qquad \mathfrak{C} \in \{\varnothing, m, 7, 0\}$
17. $\mathfrak{X}^{7}_{\mathfrak{M}} \rightarrow V/\mathfrak{X}^{m}_{\mathfrak{M}}\ \mathfrak{X}^{7}_{\mathfrak{M}} \qquad \mathfrak{X} \in \{I, \flat II, II, \ldots, VII\}$

Besides these basic elements of tonal harmony, we distinguish various scale-degree substitutions and transformations. For this we introduce the function $V/\mathfrak{X}$ which transposes an arbitrary scale degree $\mathfrak{X}$ a fifth up. Herewith, Spec. 16 accounts for the classical preparation of a scale degree by its secondary dominant, stating that every scale degree, independently of its mode, chord class, and root interval, can be preceded by a chord of the dominant class, one fifth up. In the case of chord class, we denoted the independence of a chord class with the chord class variable $\mathfrak{C}$. The empty symbol, $\varnothing$, represents the major class (no superscript). (Note that in all specifications that contain transposition functions, terms like $Y/\mathfrak{X}^{\mathfrak{C}}_{\mathfrak{M}}$ should always be interpreted as $(Y/\mathfrak{X})^{\mathfrak{C}}_{\mathfrak{M}}$.) Similarly, every dominant scale degree can be prepared with the minor chord one fifth above (Spec. 17). These two specifications together allow for the derivation of the typical and prominently present ii-V-I progressions in jazz harmony. (The ii-V-I progression is a very common cadential chord progression in jazz harmony, but it also occurs in other musical genres. The I-chord can practically be an arbitrary chord in a sequence which is preceded by its relative secondary

dominant a fifth interval up, the V, and its relative subdominant another fifth interval up, the ii.) The explicit annotation of these structures is very common in jazz analysis and education (see, for instance, Jaffe 1996), and also Rohrmeier (2007, 2011) incorporates Spec. 16 in his model. Spec. 16–17 interfere with Spec. 4–11, however, causing multiple analyses. Because we prefer, e.g., a II^m, V^7, and I to be explained as *Sub*, *Dom*, and *Ton*, we constrain the application of Spec. 16 and 17 to the cases where Spec. 4–11 do not apply. The rationale behind this choice is more of a pragmatic than of a musical nature, and the user of the model might also decide on a different specification preference.

Diatonic Chains of Fifths

18. $\mathfrak{X}^m_{\text{Maj}} \rightarrow Vd/\mathfrak{X}^m_{\text{Maj}} \quad \mathfrak{X}^m_{\text{Maj}}$
19. $\mathfrak{X}_{\text{Min}} \rightarrow Vd/\mathfrak{X}_{\text{Min}} \quad \mathfrak{X}_{\text{Min}}$

The model also accounts for the diatonic chains of fifths in major (Spec. 18) and minor (Spec. 19) keys. These diatonic-chain specifications are necessary to explain the typical cycle-of-fifths progressions: $I\ IV\ VII^m\ III^m\ VI^m\ II^m\ V^7\ I$ (see Figure 2 for the major case, as well as the *Autumn Leaves* example in the next section). These strong chord progressions are very commonly found throughout Western tonal music. Also, Rohrmeier's (2011) model has a rule dedicated specifically to this progression. To reduce multiple ambiguous solutions, we constrain the major-key specification to apply only to the minor chords, because I, IV, and V^7 translate directly to *Ton*, *Sub*, and *Dom*, respectively. Similarly, Spec. 19 captures the same phenomenon in a minor key. Here, we restrict the application of the specification only to the major chords because, again, I^m, IV^m, and V^7 translate directly to *Ton*, *Sub*, and *Dom* in minor.

We implemented one exception to the diatonic minor specification that allows the $\flat VI$ to precede the II^m in minor. Here, the major chord precedes a minor chord. See the $\text{E}\flat^{\Delta}$ $\text{Am}^{7\flat5}$ in the analysis of *Autumn Leaves*, later in this article.

Tritone Substitution

20. $\mathfrak{X}^7_{\mathfrak{M}} \rightarrow \flat V/\mathfrak{X}^7_{\mathfrak{M}}$

The harmony model in HarmTrace allows for various scale-degree transformations. Every chord of the dominant class can be transformed into its tritone substitution with Spec. 20. This specification uses another transposition function $\flat V/\mathfrak{X}$ which transposes a scale degree $\mathfrak{X}$ a diminished fifth—a tritone—up (or down). The tritone substitution specification allows for the derivation of progressions with a chromatic baseline, e.g., Am $\text{G}\sharp^7$ G, and is very common in jazz harmony (see, for instance, Jaffe 1996). Also, a tritone specification was earlier used in a similar model in De Haas et al. (2009). Because we want the application of the Spec. 16–20 to terminate, we limit the number of possible recursive applications of these rules (see the Parsing section).

Diminished Sevenths

21. $\mathfrak{X}^0_{\mathfrak{M}} \rightarrow \flat III/\mathfrak{X}^0_{\mathfrak{M}}$
22. $\mathfrak{X}^7_{\mathfrak{M}} \rightarrow II\flat^{7\flat9}/\mathfrak{X}^0_{\mathfrak{M}}$

Diminished seventh chords can have various roles in tonal harmony. An interesting characteristic of these chords—consisting only of notes separated by minor third intervals—is that they are completely symmetrical. Hence, a diminished seventh chord has four enharmonic equivalent chords that can be reached by transposing the chord by a minor third with the transposition function $\flat III/\mathfrak{X}$ (Spec. 21). In general, we treat diminished seventh chords as dominant-seventh chords with a $\flat$9 and no root note, which is a common interpretation of this chord (Saslaw 2013). For instance, in a progression Am^7 $\text{A}\flat^0$ G^7, the $\text{A}\flat^0$ closely resembles the $\text{G}^{7\flat9}$, because a $\text{G}^{7\flat9}$ chord consists of G, B, D, F, and A$\flat$, and an $\text{A}\flat^0$ chord consists of A$\flat$, B, D, and F. This similarity is captured in Spec. 22, where $\flat II/\mathfrak{X}$ transposes a scale degree one semitone up. Similarly, by combining secondary dominants (Spec. 16) with Spec. 22, e.g., F $\text{E}\flat^0$ ($\approx \text{D}^{7\flat9}$) G, and an application of Spec. 21, e.g., F $\text{F}\sharp^0$ ($\approx\text{E}\flat^0$) G, we can account for most of the ascending diminished chord progressions. Within harmony theory, this phenomenon is usually labeled as $VII/\mathfrak{X}$, where $\text{F}\sharp^0$ would be the VII/V (in C major) in the previous example. Because our model can explain it without

a separate $VII/\mathfrak{X}$ specification, there is no need to add one.

Parallel Keys and the Neapolitan

23. $Func_{\text{Maj}} \rightarrow Func_{\text{Min}}$
24. $Func_{\text{Min}} \rightarrow Func_{\text{Maj}}$
25. $Sub_{\text{Min}} \rightarrow \flat II_{\text{Min}}$

We support borrowings from the parallel key by changing the mode but not the root of the key in the $Func_{\mathfrak{M}}$ datatype (Spec. 23 and 24). Although the parallel keys are often considered rather distantly related (there is a difference of three accidentals in the key signature), borrowing from minor in major occurs frequently in jazz harmony, and sometimes in classical music. These specifications account, for instance, for the picardy third—ending a minor piece on a major tonic. The actual implementation of Spec. 23 and 24 differs marginally to overcome endless recurring transitions between major and minor. Finally, the Neapolitan chord $\flat II_{\text{Min}}$ is categorized as being part of a Sub_{Min} structure (Spec. 25), which is also reachable in a major key through Spec. 23. Although it might be considered an independent musical event, it often has a subdominant function (Drabkin 2013).

The datatype specifications that we have presented in this section match the Haskell code closely. Nevertheless, to maintain clarity, some minor implementation details were omitted; these can be found in the real datatype specifications of the model, i.e., the Haskell code.

Modulation

HarmTrace supports chord borrowing from parallel keys, and its secondary dominant specifications can be used to explain local key changes, but it does not support modulating to all possible keys. Although adding a modulation specification to the model is straightforward, this would quickly lead to an avalanche of ambiguous analyses. For example, all specifications of the model are currently parametrized by a mode, which can be major, minor, or mode agnostic. Extending this parameter to contain the key of the piece, in line with Rohrmeier's (2011) model, is problematic: even with a constrained modulation specification that allows modulation only to specific other keys, and restricts the number of modulations, the total number of ambiguous analyses quickly explodes, given the rules of the previous section. If a user wishes to have such a modulation specification (and this is very well possible from a specification point of view), we recommend using a much smaller model. Such a model should not, for instance, contain rules for explicitly labeling chains of secondary dominants, but a secondary dominant would be analyzed as a local modulation (like Schenkerian tonicization). Even with a much smaller model, however, keeping the number of different analyses under control might be challenging. Hence, we chose to first explore the usability of HarmTrace without modulation, keeping the model fast and flexible.

Another, more practical, solution is using external key information, either obtained from key signatures in the score, or by applying an automatic key-finding algorithm. Within the audio domain, key-finding algorithms have improved considerably in the last few years, providing annotations of both the global key and key changes. Within the symbolic domain, the key is often annotated explicitly or can be derived using symbolic key-finding algorithms (e.g., Temperley 2001). Information about the key can then be used to segment the chord sequences into sections that contain only a single key, and HarmTrace can be applied to these sections individually. For instance, in earlier work we applied HarmTrace to improve harmonic similarity estimation, using information about key that was available in the data (De Haas et al. 2011). We have also used HarmTrace for a chord transcription task, using a key-finding algorithm (De Haas, Magalhães, and Wiering 2012). The key-finding algorithm used in this latter work was similar to that used by Temperley (2001, ch. 7). Various different approaches to key finding exist (for a fairly recent overview, see Noland and Sandler 2009). These concrete applications of HarmTrace show that the availability of key data does not have to hamper practical use.

Parsing

Once we have a formal specification as a datatype, the next step is to define a parser that transforms textual chord labels into values of our datatype. Writing a parser that parses labels into our datatype would normally mean writing tedious code that closely resembles the datatype specification. In Haskell however, we can use datatype-generic programming techniques (Jeuring et al. 2009) to avoid writing most of the repetitive portions of the code. (These techniques are not to be confused with regular polymorphism, as in Java generics.) Moreover, we derive not only the parser automatically, but also a pretty-printer for displaying the harmony analysis in tree form, and functions for comparing these analyses. This makes the development and fine-tuning of the model much easier, as only the datatypes have to be changed, and the code adapts itself automatically. The technical details of the implementation of our model, and of the generic-programming techniques we use, are covered in an earlier publication (Magalhães and De Haas 2011).

Because music is an ever-changing, culturally dependent, and extremely diverse art form, we cannot hope to model all valid harmonic relations in our datatype. Furthermore, song data may contain mistakes or mistyped chords, perhaps feature extraction noise, or malformed data of dubious harmonic validity. In HarmTrace we use a parsing library featuring error-correction: Chords that do not fit the structure are automatically deleted or preceded by inserted chords, according to the datatype structure (Swierstra 2009).

The error-correction process uses heuristics to find a good parse tree in a reasonable amount of time. When faced with a chord that does not fit the harmony model, it will consider all possible combinations of deletion and insertion of chords (up to a fixed depth of three steps) to adapt the chord sequence to the model. In this way, the simplest corrections (involving the fewest insertions or deletions) are chosen. We could also assign different costs to each specification, in order to prefer some rules over others. In practice, we did not find this necessary. The order in which the rules are specified also matters, as earlier rules take precedence over later rules; we use this fact to guide the correction process.

For most songs, parsing proceeds with no corrections, or only very few. Songs with a very high error ratio denote multiple modulations, bad input, or a wrong key assignment. Note that, depending on the severity of "unexpectedness" of a chord, there might be multiple error-corrections necessary to create a valid analysis (e.g., one deletion and two insertions).

In our model, one particular parameter has a large influence on the parsing and error-correction process. This parameter controls the number of recursive applications of the specifications for secondary dominant and the like (Spec. 16–20). It must be set carefully, because setting it to a very low value will lead to bad analyses, and setting it to a high value will make the internally generated model very large, resulting in increased error-correction times and often sub-optimal error-correction solutions. For the examples and results in this article we have used values between five and seven.

Example Analyses

In this section we demonstrate the analytic power of the HarmTrace system. The input presented to HarmTrace consists of a sequence of plain-text chord labels in the syntax defined by Harte et al. (2005), and the output consists of a parse tree similar to those often used in natural language processing. In these trees, the input chord labels are the leaves, which are subsequently grouped into scale degrees, scale-degree transformations, and functional categories, and finally collected in a *Piece* node, as prescribed by the rules of the model. The notation used in the parse trees is identical to the notation used to describe the datatype specifications in the previous section. Musical harmony can be very ambiguous, and sometimes multiple analyses are defensible. A strength of HarmTrace is that, by changing the specifications in the previous section, it can be easily adapted to meet the musical intuitions of its users.

We start by analyzing the B section of the *Autumn Leaves* jazz standard as found in *The Real Book* (Sher 1988). The parse tree of this piece in the key of G minor is depicted in Figure 3. Within the piece, the three A^0 D^7 Gm sequences translate into subdominant, dominant, and tonic

Figure 3. The harmonic analysis of the B section of the jazz standard Autumn Leaves*, as derived by HarmTrace. For convenience, we left out the mode subscript* Min *and printed the chord labels as commonly found in lead sheets. The key of the piece is G minor. The* Δ *denotes an added major seventh.*

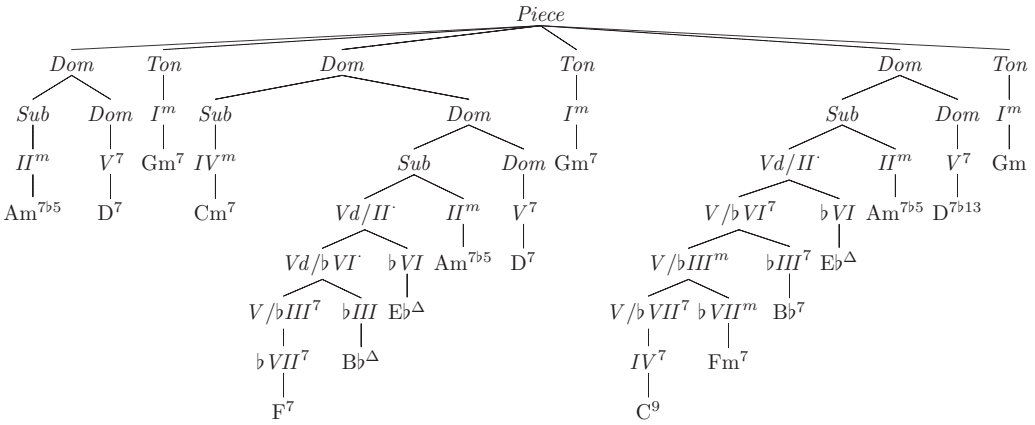


Figure 3

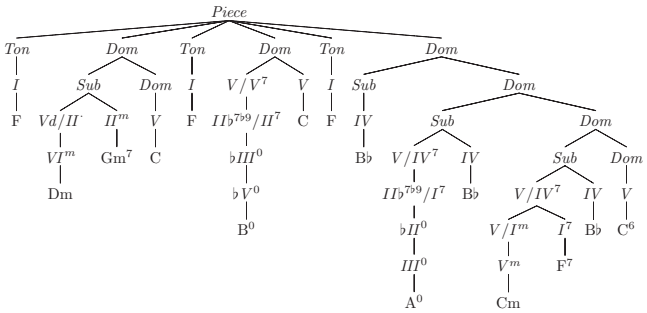


Figure 4

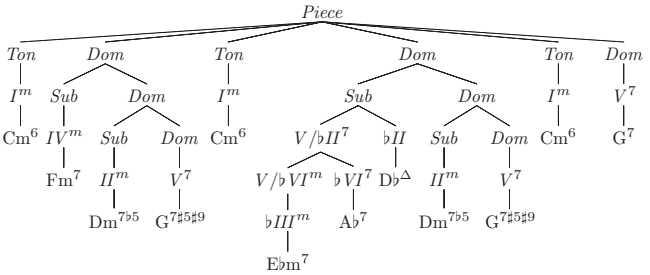


Figure 5

categories (Spec. 9, 6, and 5, respectively), forming ii-V progressions, towards the tonic of the piece. The second and third *Dom* branches of *Piece* display different types of descending fifth movements, both of which build up tension towards a D^7: The progressions from F to B♭ are labeled either as a secondary dominant or as a diatonic descending fifth, depending on whether the chord is dominant or major (Spec. 16 or 19). In the current analysis, the Cm chord is viewed only as a subdominant of the Gm chord. Additionally, the Cm could have been seen as being part of a ii-V-I motion to the $E♭^{\Delta}$ chord. HarmTrace misses this additional analysis, however, because the maximal number of recursive applications has been reached.

Although the model has a bias towards a jazz repertoire, it can be used to analyze Bach chorales as well. In Figure 4 we present the HarmTrace analysis of the first nine measures of Bach's harmonization of the chorale melody *Ich dank' dir schon durch deinen Sohn*. The key of the piece is F major. After an introduction of the tonic, a diatonic chain of fifths prepares the dominant, C, which subsequently resolves to the tonic. The next branch prepares a C with a B^0, which is VII/V. As explained in the previous section, the B^0 is enharmonically equivalent to $A♭^0$ ($III♭^0$) which is very similar to a $G^{7♭9}$ (denoted by $II♭^{7♭9}/II$, Spec. 22), which is in turn the V/V, preparing the C chord in the key of F. The final branch creates some harmonic movement around the B♭ by preparing the second B♭ with a VII/IV. The $VII/\mathfrak{X}$ derivation is identical to this one, but transposed down a whole step. The fragment concludes with a descending fifth preparation of the subdominant, followed by the dominant.

In Figure 5 we show the analysis of another well-known standard, *Blue Bossa*. The progression

Figure 6. Two analyses of a phrase from Tebe Poem *by Dmitry Bortniansky (1751–1825). We compare an analysis adapted from Rohrmeier (2011), appearing above the score (a), with an analysis produced by HarmTrace (b). The nodes labeled* TR *in Rohrmeier's analysis indicate "tonal regions," a concept not included in our model.*

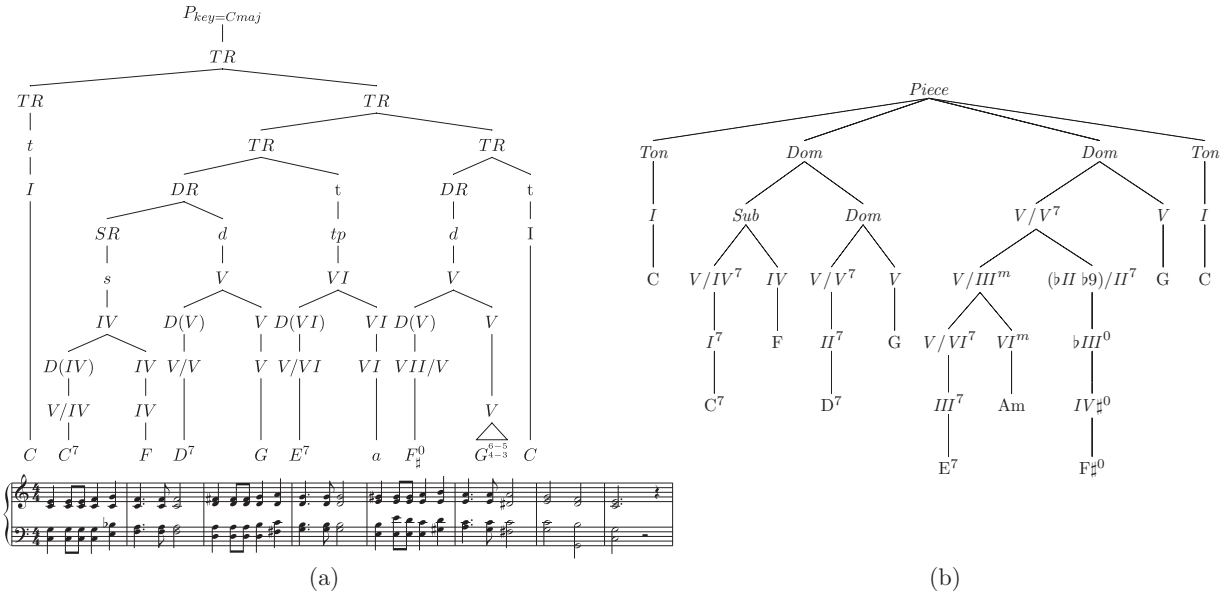


Figure 6

Figure 7. An excerpt of the analysis of It Don't Mean a Thing (If It Ain't Got That Swing).

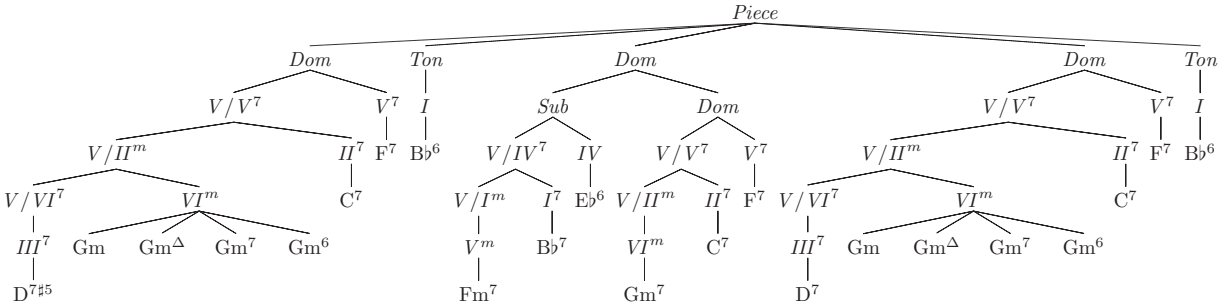


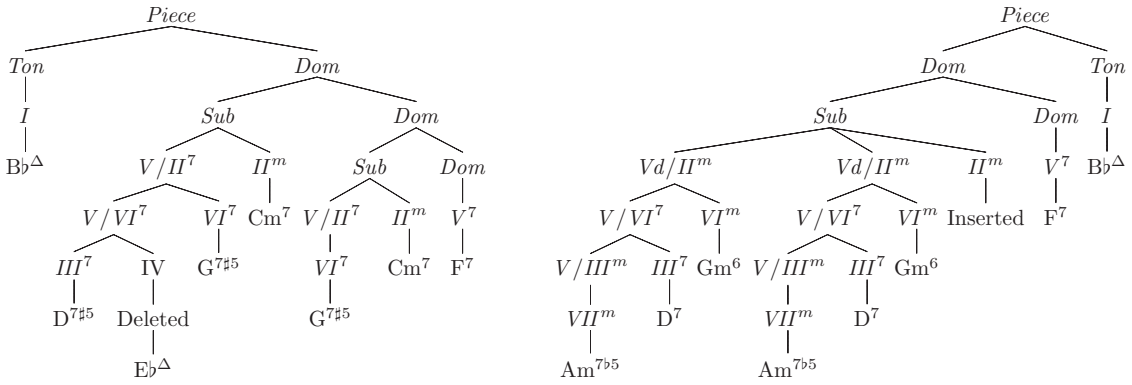
Figure 7

starts by introducing the tonic, Cm, followed by a perfect cadence. The B-part displayed in the second *Dom* branch shows a ii-V-motion to the Neapolitan $\flat II$ (Spec. 25) and is followed by a ii-V-I to Cm.

Figure 6 displays the score and two analyses of an extract from *Tebe Poem*, by Dmitry Bortniansky. The analysis in Figure 6a is the theoretical analysis proposed by Rohrmeier (2011). Although the notation used by Rohrmeier differs slightly from the notation used in this article, the analyses are clearly similar. There are also some differences. Rohrmeier connects the tonic, dominant, and subdominant nodes in tonal regions whereas HarmTrace (see Figure 6b) does not. We elaborate on this issue in the Discussion section. Another difference in derivation is that because we treat the $F\sharp^0$ ($\approx D^{7\flat 9}$) as a V/V, the E^7 and Am are analyzed as being part of a larger chain of fifths.

In Figure 7 we show the HarmTrace analysis of the jazz standard *It Don't Mean a Thing (If It Ain't Got That Swing)*. The analysis shows how similar Gm chords are grouped under one VI^m node. It

Figure 8. Two examples that illustrate error correction: excerpts from the jazz standard Someday My Prince Will Come *(a) and the B section of the standard* There Is No Greater Love *(b).*



furthermore illustrates how the *Sub* and *Dom* nodes are prepared by chains of secondary dominants.

We conclude this section with two small examples that contain error corrections. These corrections should not be interpreted as corrections of musical errors, but they illustrate the behavior of HarmTrace in cases where harmonies cannot be explained by the model. The example in Figure 8a is an excerpt of the jazz standard *Someday My Prince Will Come*, Figure 8b is taken from the jazz standard *There Is No Greater Love*. In *Someday My Prince Will Come*, the model cannot explain the $E\flat^{\Delta}$ where it occurs. Because the D^7 can immediately resolve to the G^7, the parser deletes the $E\flat^{\Delta}$. The model specification does not allow a *Sub* to translate into a VI^m scale degree. Adding such a specification would cause a large number of ambiguous solutions, if the diatonic fifth specification (Spec. 19) were not constrained. Therefore, in Figure 8b, the model needs a diatonic chain of fifths to explain the VI^m and the parser solves this by inserting a II^m. Corrections like the ones in Figure 8 represent typical examples of error corrections in HarmTrace.

Experimental Results

To demonstrate that HarmTrace can be efficiently and effectively used in practice, we evaluate its parsing performance on two chord sequence data sets: a small data set we have used in earlier work (De Haas et al. 2009), and a larger data set (used in De Haas, Veltkamp, and Wiering 2013). We refer to these two data sets as `small` and `large`, respectively. The `small` data set contains 72 chord sequences that describe mainly jazz pieces. The `large` data set contains 5,028 chord sequences that describe jazz, Latin, and pop pieces, as well as a few classical works. Both data sets consist of textual chord sequences extracted from user-generated Band-in-a-Box files that were collected on the Internet. (Band-in-a-Box [Gannon 1990] is a commercial software package that generates accompaniment, given a chord sequence.) For the extraction of the plain-text chord labels we have extended software developed by Mauch et al. (2007). To our knowledge, the `large` data set is the largest data set of symbolic chord sequences available to the research community, as of June 2013.

The `small` data set contains a selection of pieces that were checked manually and "make sense" harmonically, and the `large` data set includes many songs that are harmonically atypical. This is because the files are user-generated, so they contain peculiar and unfinished pieces, wrong key assignments, and other errors; it can therefore be considered "real world" data. Also, the `large` data set contains pieces that modulate, and even some pieces that might be considered atonal (e.g., John Coltrane's *Giant Steps*). We deliberately chose to use a "real

Table 1. Parsing Results

Data Set	*Deletions*	*Insertions*	*Corrections*	*Chords*	*Parse Time*	*Total Time*
`small`	0.83	2.79	3.63	42.49	10.00	0.72
`large`	3.38	9.85	13.24	62.05	76.53	384.81

Numbers of deletions, insertions, and total corrections per song; total number of chords per song; parsing time per song (in msec); total parsing time (in sec).

world" data set to show that HarmTrace is robust against noisy data, offers good performance in terms of parsing speed, and still delivers analyses that make sense.

Parsing Results

When parsing the data we measure the number of parsed chords, deleted chords, and inserted chords, as well as parsing time. These numbers are summarized in Table 1. Both runs were performed on the same Intel Core 2 6600 machine running at 2.4 GHz with 3 GB of random-access memory compiled using GHC version 7.0.3.

On the `small` data set the HarmTrace model performs very well. The songs are parsed quickly and, on average, fewer than one chord per song is deleted. Also, fewer than three insertions are necessary, on average, for a piece to parse. It would have been possible to adapt the model in such way that the `small` data set would parse without any errors, as was done by De Haas et al. (2009). We chose to accept this small number of error corrections and keep our grammar small and easy to comprehend. The data set is parsed within a second.

For the `large` data set, the parsing time per song increases considerably, due to a larger number ambiguous solutions and increased error-correction. The 5,028 chord sequences are still parsed reasonably quickly, in 6 min 25 sec. The number of error corrections increases considerably, but the parser never crashes or refuses to produce valid output. The higher number of error corrections is expected, because this data set contains songs with modulations, atonal harmonies, and a variety of errors. Still, HarmTrace keeps the number of deleted chords under 6 percent of the total chords parsed.

When we compare the parsing results of the `small` data set with the results of an older, Java-based parser (De Haas et al. 2009), we notice that HarmTrace is much faster. The Java-based parser took more than 9 min to parse this data set. We cannot compare the parsing results of the `large` data set because the majority of the pieces are rejected by the grammar used by the older parser. This emphasizes how important the error-correction process is. Even with error correction, HarmTrace parses the `large` data set faster than the Java-based parser parses the `small` data set.

Discussion

We have presented HarmTrace, a system that automatically analyses sequences of musical chord labels. Implementing our system in Haskell has proven to be a profitable decision, given the advantages of error-correcting parsers and datatype-generic programming. We have shown that HarmTrace can handle corpora of considerable size, parses chord progressions quickly, and is robust against noisy data. HarmTrace currently does not support full modulation, but it can explain changes between parallel major and minor keys. That the lack of full modulation does not have to hamper practical application is shown by the application of HarmTrace to practical MIR tasks like automatic harmonic-similarity estimation, chord transcription, and automatic harmonization. Although the model presented has a bias towards jazz harmony, we have shown that it can be used to analyze some classical works as well.

If we compare HarmTrace to other models of tonal harmony, we notice various differences. The theoretical work of Steedman (1984), for instance, focuses only on the structure of the (very particular) style of twelve-bar blues, whereas our model aims to formalize the core of tonal harmony with a bias towards jazz harmony, including the twelve-bar blues. Although our work draws on the work of Rohrmeier (2007, 2011), there are also considerable contrasts. The most pronounced difference from Rohrmeier's CFG is that the latter features modulation and tonicization. By tonicizing to a local tonic, chords are analyzed with respect to that local tonic. As a consequence, his approach can explain secondary dominants by tonicization, whereas the HarmTrace model uses a more jazz-oriented approach to harmony by identifying ii-V-I motions, (e.g., in Figure 3). For instance, in a progression in the key of C major, after moving to the subdominant, F can be viewed as a local tonic, allowing the derivation of Gm and C as local subdominant and local dominant. A benefit of Rohrmeier's approach is that it is also possible to derive B♭ C as a local subdominant/dominant pair. A specification for this would be easy to add to the HarmTrace model. Implementing both tonicizations and secondary dominants, as Rohrmeier suggests, would be problematic, however, because both rules explain the same phenomena. After all, the preparation of F by C can be explained both by the tonicization rules as well as by the secondary dominant rules. This would inevitably lead to an explosion of ambiguous solutions for a harmony progression featuring secondary dominants.

Another difference is that Rohrmeier groups tonics and dominants into higher-order phrases or functional regions. He acknowledges that these rules are highly ambiguous, but chooses to keep them for theoretical completeness. The problem is that the harmonic information alone generally does not provide enough information for determining phrase boundaries. For instance, it is unclear whether *Ton Dom Ton* represents a half-cadence phrase followed by a tonic (*Ton Dom*) (*Ton*), or an introduction of the tonic followed by a perfect-cadence phrase (*Ton*) (*Dom Ton*). We believe that such clusterings should be done in a post-processing step, based on metrical positions and phrase-length constraints.

On the whole, when we compare HarmTrace to other models of tonal harmony, we observe that most models remain purely theoretical. This is regrettable, because although theoretical work can yield valuable insights, having an implementation of a model allows it to be evaluated empirically and used in practice. HarmTrace has been clearly demonstrating its practical use by its application to harmonic similarity estimation, chord recognition, and automatic harmonization. As we have seen in this article, it may take state-of-the-art programming techniques to create a model that is maintainable, has expressive power, and yet remains fast. We are confident that HarmTrace will contribute to new insights in the modeling of harmonic analysis, and that it will prove useful in practical music applications.

Acknowledgments

W. Bas De Haas was funded by the Netherlands Organization for Scientific Research, NWO-VIDI grant 276-35-001. José Pedro Magalhães was supported by EPSRC grant EP/J010995/1. Frans Wiering and Remco C. Veltkamp are supported by the FES project COMMIT/. We thank Martin Rohrmeier for the inspiring discussions and ideas about modeling harmony. We thank Anja Volk and two anonymous reviewers for comments on a draft version of this article.

References

Chemillier, M. 2004. "Toward A Formal Study of Jazz Chord Sequences Generated by Steedman's Grammar." *Soft Computing* 8(9):617–622.

Choi, A. 2011. "Jazz Harmonic Analysis as Optimal Tonality Segmentation." *Computer Music Journal* 35(2):49–66.

Chomsky, N. 1957. *Syntactic Structures*. The Hague: Mouton.

Clarke, E. 1986. "Theory, Analysis, and the Psychology of Music: A Critical Evaluation of Lerdahl, F. and

Jackendoff, R., *A Generative Theory of Tonal Music*." *Psychology of Music* 14(1):3–16.
Downie, J. S. 2003. "Music Information Retrieval." *Annual Review of Information Science and Technology* 37(1):295–340.
Drabkin, W. 2013. *Neapolitan Sixth Chord*. Oxford Music Online. Oxford: Oxford University Press. Available online at http://www.OxfordMusicOnline.com/subscriber/article/grove/music/19658 (subscription required). Accessed 25 June 2013.
Gannon, P. 1990. "Band-in-a-Box." Available online at http://www.pgmusic.com/. Accessed 25 June 2013.
De Haas, W. B., J. P. Magalhães, and F. Wiering. 2012. "Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge." In *Proceedings of the International Society for Music Information Retrieval Conference*, pp. 295–300.
De Haas, W. B., R. C. Veltkamp, and F. Wiering. 2013. "A Geometrical Distance Measure for Determining the Similarity of Musical Harmony." *International Journal of Multimedia Information Retrieval* 2(3):189–202. dx.doi.org/10.1007/s13735-013-0036-6.
De Haas, W. B., et al. 2009. "Modeling Harmonic Similarity Using a Generative Grammar of Tonal Harmony." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 549–554.
De Haas, W. B., et al. 2011. "HarmTrace: Improving Harmonic Similarity Estimation Using Functional Harmony Analysis." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 66–72.
De Haas, W. B., et al. 2012. "Chordify: Chord Transcription for the Masses." Demonstration presented at the International Society for Music Information Retrieval Conference, 8–12 October, Porto, Portugal. Available online at http://dreixel.net/research/pdf/cctfm.pdf. Accessed June 2013.
Hamanaka, M., K. Hirata, and S. Tojo. 2006. "Implementing *A Generative Theory of Tonal Music*." *Journal of New Music Research* 35(4):249–277.
Harte, C., et al. 2005. "Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations." In *Proceedings of the International Symposium on Music Information Retrieval*, pp. 66–71.
Jaffe, A. 1996. *Jazz Harmony*. Rottenburg: Advance Music, 2nd edition.
Jeuring, J., et al. 2009. "Libraries for Generic Programming in Haskell." In P. Koopman, R. Plasmeijer, and D. Swierstra, eds. *Advanced Functional Programming, 6th International School, Lecture Notes in Computer Science*, vol. 5832. Berlin: Springer, pp. 165–229.
Koops, H. V., J. P. Magalhães, and W. B. de Haas. 2013. "A Functional Approach to Automatic Melody Harmonisation." In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design (FARM '13)*.
Lerdahl, F., and R. Jackendoff. 1996. *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press.
Magalhães, J. P., and W. B. de Haas. 2011. "Functional Modelling of Musical Harmony: An Experience Report." In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pp. 156–162.
Marsden, A. 2010. "Schenkerian Analysis by Computer: A Proof of Concept." *Journal of New Music Research* 39(3):269–289.
Mauch, M., et al. 2007. "Discovering Chord Idioms Through Beatles and Real Book Songs." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 255–258.
Noland, K., and M. Sandler. 2009. "Influences of Signal Processing, Tone Profiles, and Chord Progressions on a Model for Estimating the Musical Key from Audio." *Computer Music Journal* 33(1):42–56.
Pachet, F. 1999. "Surprising Harmonies." *International Journal of Computing Anticipatory Systems* 4:139–161.
Patel, A. D. 2003. "Language, Music, Syntax and the Brain." *Nature Neuroscience* 6:674–681.
Peyton Jones, S. 2003. "Haskell 98 Language and Libraries: The Revised Report." *Journal of Functional Programming* 13(1):7–255.
Piston, W. 1991. *Harmony*. London: Gollancz, 7th edition. Revised and expanded by Mark DeVoto.
Rameau, J.-P. 1722. *Traité de l'harmonie*. Paris: Ballard. Translated by P. Gossett as *Treatise on Harmony*. 1971. New York: Dover.
Riemann, H. ca. 1895. *Harmony Simplified: Or the Theory of the Tonal Functions of Chords*. London: Augener. Originally published as *Vereinfachte Harmonielehre. Oder: Die Lehre von den tonalen Funktionen der Akkorde*.
Roads, C. 1979. "Grammars as Representations for Music." *Computer Music Journal* 3(1):48–55.
Rohrmeier, M. 2007. "A Generative Grammar Approach to Diatonic Harmonic Structure." In *Proceedings of the Sound and Music Computing Conference*, pp. 97–100.
Rohrmeier, M. 2011. "Towards a Generative Syntax of Tonal Harmony." *Journal of Mathematics and Music* 5(1):35–53.
Saslaw, J. 2013. *Diminished Seventh Chord*. Oxford Music Online. Oxford: Oxford University Press. Available

online at http://www.OxfordMusicOnline.com/subscriber/article/grove/music/07806 (subscription required). Accessed 25 June 2013.
Schenker, H. 1935. *Neue musikalische Theorien und Phantasien III: Der freie Satz*. Vienna: Universal Edition.
Schrijvers, T., et al. 2009. "Complete and Decidable Type Inference for GADTs." In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pp. 341–352.
Sher, C., ed. 1988. *The New Real Book*. Petaluma, California: Sher Music.
Steedman, M. J. 1984. "A Generative Grammar for Jazz Chord Sequences." *Music Perception* 2(1):52–77.
Steedman, M. J. 1996. "The Blues and the Abstract Truth: Music and Mental Models." In J. Oakhill and A. Garnham, eds. *Mental Models in Cognitive Science*. Mahwah, New Jersey: Erlbaum, pp. 305–318.
Swierstra, S. D. 2009. *Combinator Parsing: A Short Tutorial*. Berlin: Springer, pp. 252–300.
Temperley, D. 2001. *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts: MIT Press.
Temperley, D., and D. Sleator. 1999. "Modeling Meter and Harmony: A Preference-Rule Approach." *Computer Music Journal* 23(1):10–27.
Winograd, T. 1968. "Linguistics and the Computer Analysis of Tonal Harmony." *Journal of Music Theory* 12(1):2–49.