HarmTrace: Automatic functional harmonic analysis

W. Bas de Haas

José Pedro Magalhães

Frans Wiering

Remco C. Veltkamp

Technical Report UU-CS-2011-023
July 2011

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Abstract

Music scholars have been intensively studying tonal harmony 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, which builds on well-known theories of tonal harmony. In contrast to many other theories which 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 which are uniquely available in the Haskell programming language. We show that our system is fast, easy to modify and maintain, is robust against noisy data, and returns harmonic analyses that comply with Western tonal harmony theory.

1 Introduction

For ages, musicians, composers, and musicologists have been theorizing the structure of music to better understand how music is perceived, performed, and appreciated. In particular, tonal harmony exhibits a considerable amount of structure and regularity, and the first theories describing tonal harmony date back at least to the 18th century (Rameau 1971). Since then, a rich body of literature has emerged that aims at explaining the harmonic regularities in both informal and formal models (e.g. Lerdahl and Jackendoff 1996). Such models have attracted numerous computer music researchers to automate the analysis and generation of harmony. However, most of these theories 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 analyses 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 paper we present HarmTrace[1] , a system that meets these requirements using state-of-the-art functional programming techniques. HarmTrace allows 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 modelled the core rules of Western tonal harmony as a (large) context-free grammar (CFG, Chomsky 1957). Later, De Haas et al. (2009) implemented this grammar and specifically tuned it for jazz harmony, with the aim of modelling 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 non-parsing pieces, and controlling the number of ambiguous solutions. Since it relies on advanced functional programming techniques not readily available in most programming languages, HarmTrace is inextricably bound to Haskell (Bird 1998). Haskell is a purely functional programming language with strong static typing. It is purely functional because its functions, like regular mathematical functions, guarantee producing the same output when given the same input. It is strongly

[1]Harmony Analysis and Retrieval of Music with Type-level Representations of Abstract Chord Entities

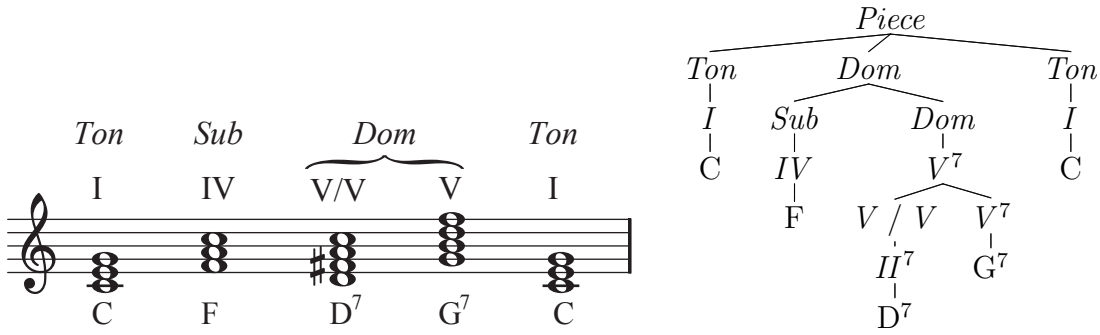

Figure 1: A typical chord sequence and its harmonic analysis (as generated by HarmTrace) The chord labels are printed below the score, and the scale degrees and functional analysis above the score. We ignored voice-leading for simplicity.

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,[2] 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.

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 and DeVoto 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, while removing other chords can significantly change how the chord sequence is perceived. This is illustrated in the sequence displayed in Figure 1: the D^7 chord in this sequence can be removed without changing the general structure of the harmony, while 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 organised hierarchically. This hierarchical organisation is reflected in the tree in Figure 1. 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.

Following 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 could imagine retrieving harmonically related pieces of music, like cover-songs, classical variations, or all blues pieces in a corpus. For performing such searches, a measure of harmonic similarity is essential. De Haas et al. (2009, 2011) and Magalhães and de Haas (2011) show that analysing the hierarchical relations between chords in a sequence significantly improves the quality of a harmonic similarity

[2] `http://www.haskell.org/ghc/`

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

A fully-functional model of tonal harmony that can quickly analyse chord sequences offers several benefits. First, 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. Second, because our system captures the global and local relations between chords, it can be used to improve systems designed for other tasks that can benefit from this contextual information, such as chord labelling systems. These systems traditionally determine the chord root and type based on the musical information (audio or notation) from within a limited time frame, without incorporating knowledge about the surrounding chord sequences. Last, HarmTrace could aid in (automatic) composition by generating sequences of chords, or by generating harmonically realistic continuations given a sequence of chords.

This paper is organised as follows. We start by discussing a relevant selection of the large body of existing literature on harmony theory and modelling in the next section. Subsequently, we explain our harmony model, and then evaluate 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 paper by discussing the limitations of our system, and pointing out the future directions of our research.

2 Related work

The nineteenth and twentieth century have yielded a wealth of theoretical models of Western tonal music; in particular, tonal harmony has been prominently researched. Most theories that describe the relationships between sequential chords capture notions of order and regularity; some combinations of chords sound natural while others sound awkward (e.g. Rameau 1971). These observations led music theorists to develop ways to analyse the function of a chord in its tonal context (e.g. Riemann 1893). 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 paper.

Seminal has been the *Generative Theory of Tonal Music* (GTTM, Lerdahl and Jackendoff 1996) that further formalised the ideas of Schenker (1935). Lerdahl and Jackendoff structured 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 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 et al. 2006).

The recursive formalisation proposed by Lerdahl and Jackendoff suggests a strong connection between language and music. Also, many other authors have argued that tonal harmony should be organized in hierarchical way similar to language, leading to numerous linguistically-inspired models since the 1960's (Roads 1979). One of the pioneers to propose a grammatical approach to harmony was Winograd (1968). More recently, Steedman (1984, 1996) modelled the typical four-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 analysing 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 generative formalism proposed by Rohrmeier (2007, 2011), which the HarmTrace model greatly draws on, expands these earlier approaches in a number of ways. Rohrmeier gives an encompassing account of how tonal harmony relationships can be modelled using a generative CFG with variable binding. It aims to model form, phrasing, theoretical harmonic function (Riemann 1893), scale degree prolongation (Schenker 1935; Lerdahl and Jackendoff 1996), and modulation. Rohrmeier's grammar differs from earlier grammatical formalisms in various ways. Steedman's approach (Steedman 1984, 1996) merely concerns blues progressions. It features seven context-sensitive rules (with variations), but it omits a number of theoretically important features to support broader domains. Rohrmeier's formalism also differs from GTTM: GTTM aims at describing the core principles of tonal cognition, and harmony is covered mainly as a prolongational phenomenon, while Rohrmeier's formalism describes the structure of tonal harmony from an elaborate 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 used context-free parsing techniques hampered both theoretical as well as practical improvements. First, a sequence of chords that does not match the context-free specification precisely is rejected and no information is given to the user. For example, appending one awkward chord to an otherwise grammatically correct sequence of chords forces the parser to reject the complete sequence, not returning any partial information about what it has 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 grammatical models discussed above. A major drawback of context-free grammars 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 used rules. However, this does not overcome the fact that, for some pieces, the number of parse trees grows exponentially, given the number of input chords. Last, writing context-free rules by hand is a tedious and error-prone enterprise, especially since the grammatical models can become rather large. For instance, a rule generalising 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 generalising 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 formalise the hierarchical relations between chords. Our system continues where Temperley's left off: we assume we have the chord and key information of the piece, and model the global and local dependency relations between these chords. Hence, the input to our model consists of plaintext key and chord label information.

3 The HarmTrace system

In this section we explain how we model the regularities and hierarchical dependencies of tonal harmony. HarmTrace transfers the ideas of De Haas et al. (2009) to a functional setting. While 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. However, this comes at a price: our present model does not support full modulation, and can only distinguish between parallel keys—going from major to minor or vice versa without changing the root of the key. As a consequence, this requires the model to have information about the key of the piece. Also, since we mainly use jazz-oriented input data in this article, we also include some specific jazz harmony specifications. Figure 2 shows an example analysis as produced by HarmTrace. The chords 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. However, in most of the ambiguous cases it is clear from the context which of the possible solutions is the preferred one. Hence, we can select the favoured analysis by constraining the application of the specification leading to the undesired analysis. 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 (generalised) algebraic data types and context-free production rules. A CFG defines a language: given a set of production

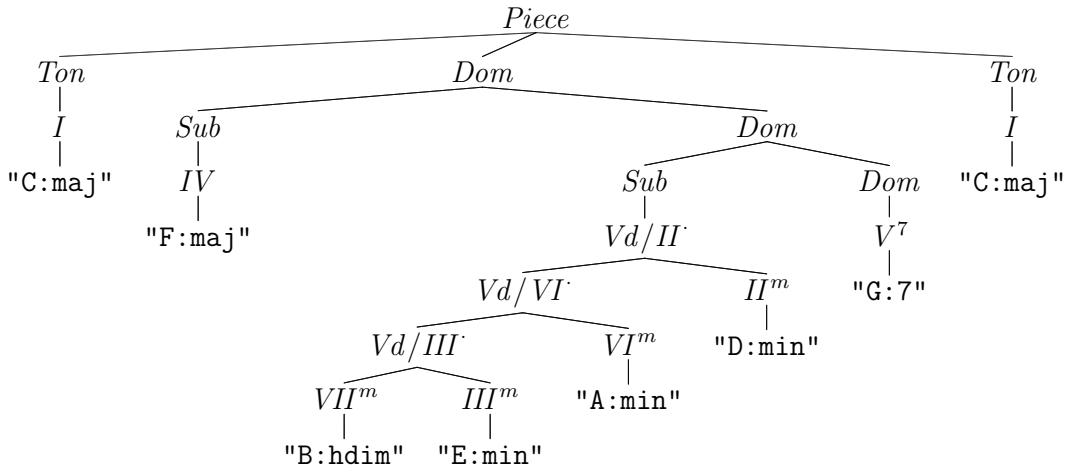

Figure 2: An example of a diatonic cycle of fifths progression in C major. The leaves represent the input chords and the internal nodes denote the harmony structure. *Ton*, *Dom*, and *Sub* denote tonic, dominant, and subdominant. The Vd/X nodes represent diatonic fifth successions.

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. Similar 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. However, an important difference between a CFG and a Haskell datatype is that datatypes provide more much modelling freedom and control, especially the *generalised* algebraic datatypes (Schrijvers et al. 2009) that we use. They allow constraining the number of applications of a specification, constraining the conditions for application, and ordering specifications by their importance. This allows defining mode and key-specific specifications, excluding scale degree-specific applications (e.g. Spec. 18) of transposition functions, and preferring certain specifications over others. For technical details, we refer to Magalhães and de Haas (2011) and the code online (package `HarmTrace-0.6`).

3.1 A model of tonal harmony

We now elaborate on how our harmony datatypes are organised. 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. We start by introducing a variable (denoted with bold font) **m** for the mode of the key of the piece, which can be major or minor. The mode variable is used to parametrise all the specifications of our harmonic datatype specification; some specifications hold for both modes (**m**), while other specifications hold only for the major (Maj subscript) or minor mode (Min). Similar to

a CFG, we use a | to denote alternatives, and a + to represent optional repetitions of a datatype.

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

Spec. 1–3 define that a valid chord sequence, $Piece_{\mathbf{m}}$, consists of at least one and possibly more functional categories. A functional category classifies chords as being part of a tonic ($Ton_{\mathbf{m}}$), dominant ($Dom_{\mathbf{m}}$), or subdominant ($Sub_{\mathbf{m}}$) structure, where a subdominant must always precede a dominant. 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 order of the dominants and tonics is not constrained by the model, and they are not grouped into larger phrases.

4 $Ton_{\text{Maj}} \rightarrow I_{\text{Maj}} \mid I_{\text{Maj}}\ IV_{\text{Maj}}\ I_{\text{Maj}} \mid III^{m}_{\text{Maj}}$
5 $Ton_{\text{Min}} \rightarrow I^{m}_{\text{Min}} \mid I^{m}_{\text{Min}}\ IV^{m}_{\text{Min}}\ I^{m}_{\text{Min}} \mid III\flat_{\text{Maj}}$
6 $Dom_{\mathbf{m}} \rightarrow V^{7}_{\mathbf{m}} \mid V_{\mathbf{m}}$
7 $Dom_{\text{Maj}} \rightarrow VII^{m}_{\text{Maj}} \mid VII^{0}_{\text{Maj}}$
8 $Dom_{\text{Min}} \rightarrow VII\flat_{\text{Min}}$ $\quad \mathbf{c} \in \{\varnothing, m, 7, 0\}$
9 $Sub_{\mathbf{m}} \rightarrow II^{m}_{\mathbf{m}}$
10 $Sub_{\text{Maj}} \rightarrow IV_{\text{Maj}} \mid III^{m}_{\text{Maj}}\ IV_{\text{Maj}}$
11 $Sub_{\text{Min}} \rightarrow IV^{m}_{\text{Min}} \mid III\flat_{\text{Min}}\ IV_{\text{Min}}$

Spec. 4–11 translate the tonic, dominant, and subdominant datatypes into scale degree datatypes. 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^{Min} 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.

A scale degree datatype is parametrised 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 categorises scale degrees as one of four types of chords (denoted with superscripts) and is used to constrain the application of certain specifications, e.g. Spec. 16–17. 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 possible altered or non-altered additions, except for the diminished seventh. Chords categorised as major contain a major triad and can be extended by with non-altered additions, with exception of the dominant seventh chord

(with additions). Chords of the dominant class have a major or augmented triad and a minor seventh and can be extended with altered or non-altered notes. Finally, the diminished class contains only the diminished seventh chord. In case a specification holds for all chord classes, the chord class variable **c** is used. This allows us to define certain specifications that only hold for dominant chords, while other specifications might hold only for minor chords, etc.

Tonics can furthermore initiate a plagal cadence (Spec. 4–5). We deliberately chose not to model the plagal cadence with scale degrees and not with *Sub* and *Ton* because this keeps the number of possible analyses smaller. Also a $Sub_{\mathbf{m}}$ can translate into the a $II^{m}_{\mathbf{m}}$, because of its preparatory role, and the dominant translates into the seventh scale degree, VII^{7}_{Maj} (and $VII\flat^{7}_{\text{Min}}$ in minor) Similarly, we could have chosen to model the $Sub_{\mathbf{m}}$ to translate also to the VI^{m}_{Maj} (VIb_{Maj} in minor). However, we chose to solve this by creating specifications for chains of diatonic fifths instead (Spec. 18–19, see for instance Figure 2).

The $Ton_{\mathbf{m}}$ resolving into III^{m}_{Maj} (and $III\flat_{\text{Min}}$ in minor), is perhaps the most unexpected transformation. 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). However, in certain progressions it cannot be explained by any of these specifications, and is best assigned a tonic function since it has two notes in common with the tonic and does not really create any harmonic tension.

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 $VII^{0}_{\mathbf{m}} \rightarrow$ `"B:dim(bb7)"`

Finally, scale degrees are translated into the actual surface chords that 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 et al.'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 treat repeating chords in a natural way, we cluster chords with the same class and same scale degree, e.g. `"C:min7" "C:min9"`, in one datatype.

16 $X^{\mathbf{c}}_{\mathbf{m}} \rightarrow V/X^{7}_{\mathbf{m}}\ X^{\mathbf{c}}_{\mathbf{m}}$ $\qquad \mathbf{c} \in \{\varnothing, m, 7, 0\}$

17 $X^{7}_{\mathbf{m}} \rightarrow V/X^{m}_{\mathbf{m}}\ X^{7}_{\mathbf{m}}$ $\qquad X \in \{I, II\flat, II, \ldots, VII\}$

Besides these basic elements of tonal harmony, we distinguish various scale degree substitutions and transformations. For this we introduce the function V/X which transposes an arbitrary scale degree 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 classs, one fifth up. 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 motions in jazz harmony. However, these specifications interfere with Spec. 4–11, causing multiple ambiguous 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 were Spec. 4–11 do not apply.

18 $X^m_{\text{Maj}} \rightarrow V/X^m_{\text{Maj}}\ X^m_{\text{Maj}}$
19 $X_{\text{Min}} \rightarrow V/X_{\text{Min}}\ X_{\text{Min}}$

The model also accounts for the diatonic chains of fifths in major (Spec. 18) and minor (Spec. 19) keys. [3] 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, and the *Autumn leaves* example of the next section in Figure 3). We constrain the major key specification to only apply 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. Without these restrictions the parser would generate multiple ambiguous solutions.

20 $X^7_{\mathbf{m}} \rightarrow V\flat/X^7_{\mathbf{m}}$

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 $V\flat/X$ which transposes a scale degree X a diminished fifth—a tritone—up (or down). This tritone substitution rule allows for the derivation of progressions with a chromatic baseline, e.g. Am $G\sharp^7$ G. Because we want the application of the Spec. 16–20 to terminate, we limit the number of possible recursive applications of these rules (see parsing section below).

21 $X^0_{\mathbf{m}} \rightarrow III\flat/X^0_{\mathbf{m}}$
22 $X^7_{\mathbf{m}} \rightarrow II\flat/X^0_{\mathbf{m}}$

Diminished seventh chords can have various roles in tonal harmony. An exceptional 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 $III\flat/X$ (Spec. 21). In general, we treat diminished chords as dominant-seventh chords with a ♭9 and no root note. For instance, in a progression Am^7 $A\flat^0$ G^7, the $A\flat^0$ closely resembles the $G^{7\flat9}$, because a $G^{7\flat9}$ chord consists of G, B, D, F, and an $A\flat^0$ chord consists of A♭, B, D, and F. This similarity is captured in Spec. 22, where $II\flat/X$ transposes a scale degree one semitone up. Similarly, by combining secondary dominants

[3] We implemented one exception to the diatonic minor specification that allows the $VI\flat$ to precede the II^m in minor. Here, the major chord precedes a minor chord. See $E\flat^{\Delta}$ $Am^{7\flat5}$ in Figure 3.

(Spec. 16) with Spec. 22, e.g. F $\mathrm{E}\flat^0$ ($\approx \mathrm{D}^{7\flat 9}$) G, and an application of Spec. 21, e.g. F $\mathrm{F}\sharp^0$ ($\approx \mathrm{E}\flat^0$) G, we can account for most of the ascending diminished chord progressions. Within harmony theory this phenomenon is usually labelled as VII/X, where $\mathrm{F}\sharp^0$ would be the VII/X (in C major) in the previous example. However, since our model can explain it without a specific VII/X specification, there is no need to add one.

23 $Func_{\mathrm{Maj}} \rightarrow Func_{\mathrm{Min}}$

24 $Func_{\mathrm{Min}} \rightarrow Func_{\mathrm{Maj}}$

25 $Sub_{\mathrm{Min}} \rightarrow II\flat_{\mathrm{Min}}$

We support borrowings from the parallel key by changing the mode but not the root of the key in the $Func_{\mathbf{m}}$ 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 occur frequently in jazz harmony. 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 $II\flat_{\mathrm{Min}}$ is categorised 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.

The datatype specification 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 as found online (package `HarmTrace-0.6`).

3.2 Parsing

Having a formal specification as a datatype, the next step is to define a parser to transform 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. However, in Haskell we can use *datatype-generic programming*[4] (Jeuring et al. 2009) techniques to avoid writing most of the repetitive portions of the code. 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. For technical details of the implementation and the generic programming techniques we refer to 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, songs 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

[4]Not to be confused with regular polymorphism, as in Java generics.

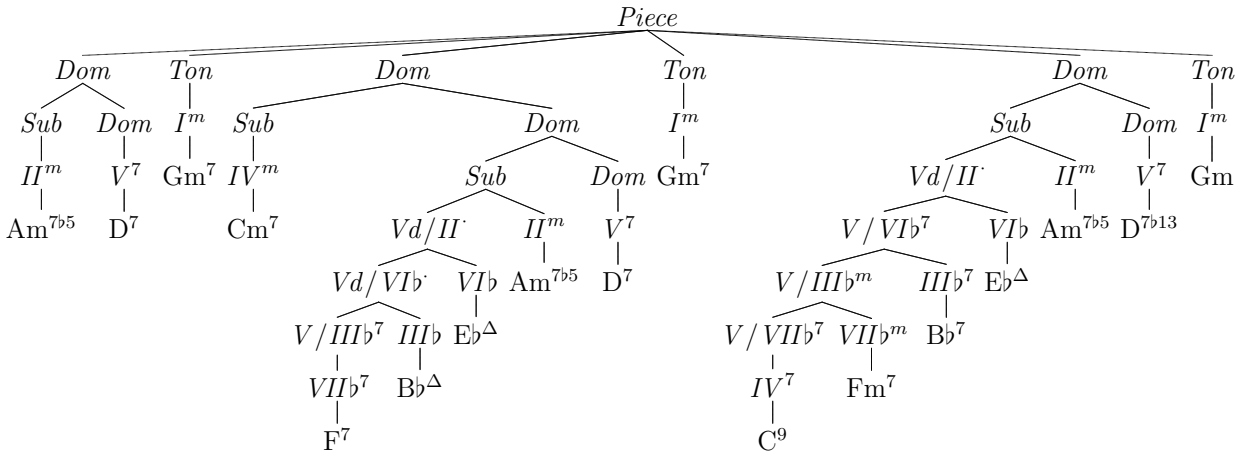

Figure 3: The harmony analysis of the B-part of the jazz standard *Autumn Leaves* as derived by HarmTrace. For convenience we left out the mode subscript Min and we printed the chord labels as is common in Real Book lead-sheets. The key of the piece is G minor.

inserted chords, according to the datatype structure (Swierstra 2009). The error-correction process uses heuristics to find a reasonable parse tree in a reasonable amount of time. For most songs, parsing proceeds with none or very few corrections. 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, while 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 paper we have used values between 5 and 7.

4 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, 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.

We start by analysing the B-part of the *Autumn Leaves* jazz standard as found in the *Real*

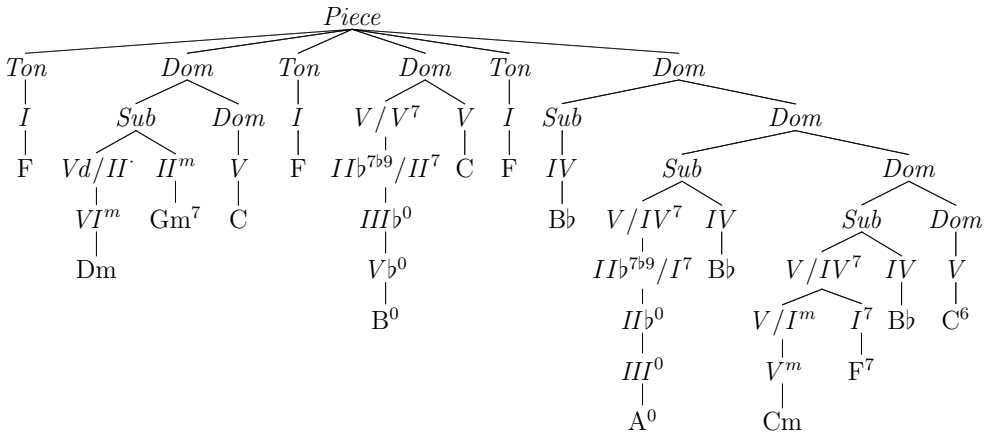

Figure 4: The automatic analysis of the first two phrases of J.S. Bach's *Ach Herre Gott, mich treibt die Not*, BWV 349, in the key of F.

Book (Various authors 2005). 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 resolve into subdominant, dominant, and tonic 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, which build up tension towards a D^7: the preparation of the B♭ by an F is labelled as a secondary dominant or as a diatonic decending fifth, depending on whether the chord is dominant or major (Spec. 16 or 19).

Although the model has a bias towards a jazz repertoire, it can be used to analyse Bach chorales as well. In Figure 4 we present the HarmTrace analysis of the first 9 measures of Bach's *Ach Herre Gott, mich treibt die Not* chorale. 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 enharmonic 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 of C. The

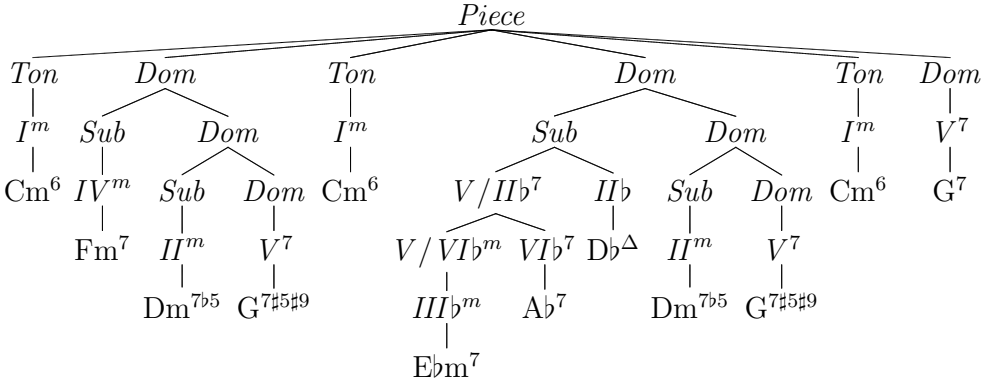

Figure 5: An analysis of *Blue Bossa* in C minor.

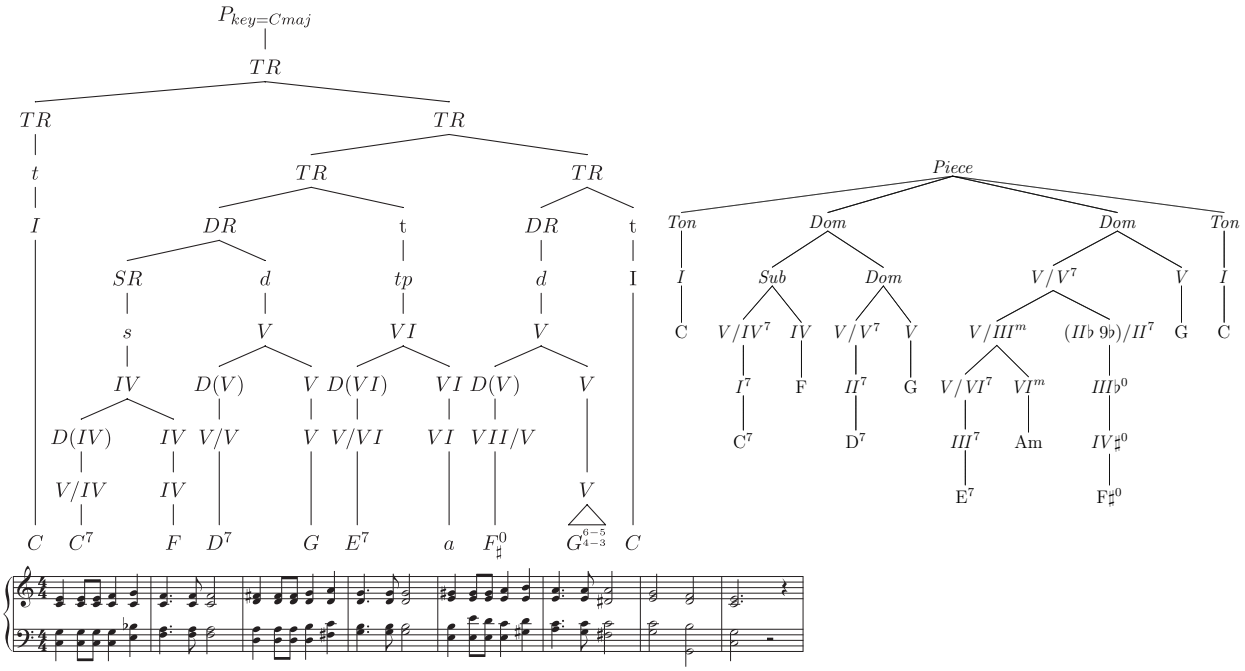

Figure 6: Two analyses of a phrase of Bornianski's piece *Tebe Poëm*. The left analysis above the score is adopted from Rohrmeier (2011), and the right analysis is the analysis output by HarmTrace.

final branch creates some harmonic movement around the B♭ by preparing the second B♭ with a VII/IV. The VII/X derivation is identical to the one explained above. The fragment is concluded with a descending fifth preparation of the subdominant that is followed by the dominant.

In Figure 5 we show the analysis of another well-known standard, *Blue Bossa*. The progression 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 $II♭$ (Spec. 25) and is followed by a ii-V-I to Cm.

Figure 6 displays the score and two analyses of Bortnianski's piece *Tebe Poëm*. The analysis on the left is the theoretical analysis proposed by Rohrmeier (2011). Although

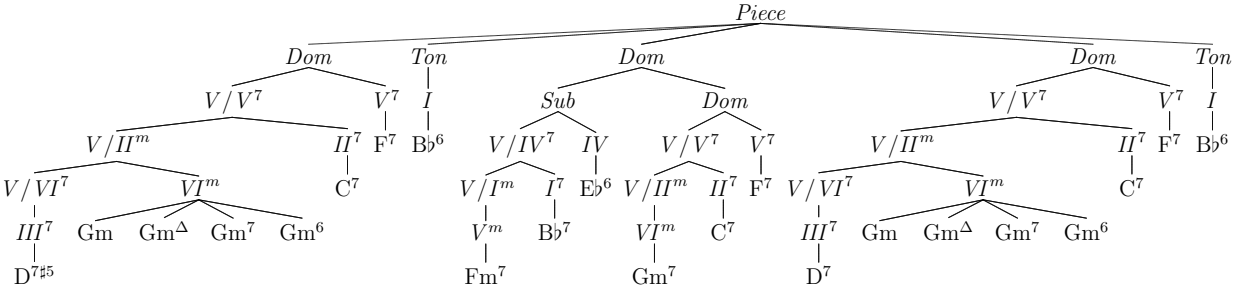

Figure 7: An excerpt of the analysis of *It don't mean a thing (if it ain't got that swing)*.

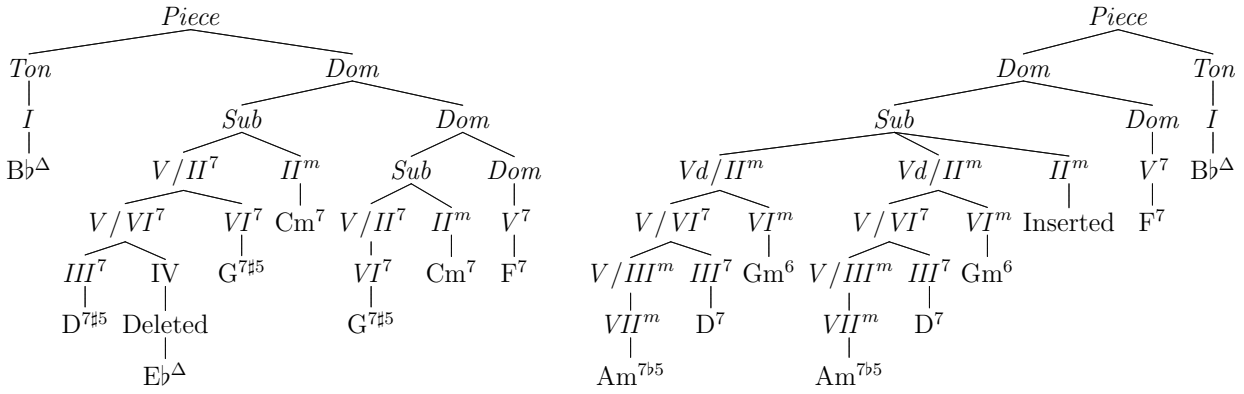

Figure 8: Two examples that illustrate error correction. On the left an excerpt of the jazz standard *There is no greater love* and on the right an excerpt of the jazz standard *Someday my prince will come* is displayed.

the notation used by Rohrmeier differs slightly from the notation used in this paper, the analyses are clearly similar. However, there are also some differences. Rohrmeier connects the tonic, dominant, and subdominant nodes in tonal regions while HarmTrace does not (we elaborate on this issue in the discussion section). Another difference in derivation is that because we treat the $\mathrm{F}\sharp^0$ ($\approx \mathrm{D}^{7\flat 9}$) as a V/V, the E^7 and Am are analysed 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 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. The example on the left in Figure 8 is an excerpt of the jazz standard *There is no greater love*, and the example on the right is an excerpt of the jazz standard *Someday my prince will come*. In the left example the model cannot explain the $\mathrm{E}\flat^{\Delta}$ at that position. Because the D^7 can immediately resolve to the G^7, the parser deletes the $\mathrm{E}\flat^{\Delta}$. 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) is not constrained. Therefore, in the example in Figure 8 on the right, 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.

5 Experimental results

To demonstrate that the HarmTrace system can be efficiently and effectively used in practice, we evaluate its parsing performance on two chord sequence datasets: a small dataset that has been used before by De Haas et al. (2009, which we will refer to as `small`), and a large dataset that is used by De Haas et al. (2010, `large`). The `small` dataset contains 72

chord sequences that describe mainly jazz pieces. The `large` dataset contains 5028 chord sequences that describe jazz, latin, pop pieces, and a few classical works. Both datasets 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 of Mauch et al. (2007). To our knowledge the `large` dataset is currently the largest dataset of symbolic chord sequences currently available to the research community.

The `small` dataset contains a selection of pieces that were checked manually and "harmonically make sense", while the `large` dataset includes many songs that are harmonically atypical. This is because the files are user-generated, contain peculiar and unfinished pieces, wrong key assignments, and other errors; it can therefore be considered "real life" data. Also, the `large` dataset contains pieces that modulate, and even some pieces that might be considered atonal, e.g. *Giant Steps*. We deliberately chose to use a "real life" dataset to illustrate that HarmTrace is robust against noisy data, offers good performance in terms of parsing speed, and still delivers analyses that make sense.

5.1 Parsing results

When parsing the data we measure the number of parsed chords, deleted chords, inserted chords, and 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` dataset 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 for a piece to parse, on average. It would have been possible to adapt the model in such way that the `small` dataset would parse without any errors, as was done by De Haas et al. (2009). However, we chose to accept this small number of error corrections and keep our grammar small and easy to comprehend. The dataset is parsed within a second.

For the `large` dataset the parsing time per song increases considerably, mostly because the ambiguity of our model can make the error-correction process rather expensive. However, the 5028 chord sequences are still parsed reasonably fast, in 6 min 24 s. 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, since this dataset contains songs

Dataset	del/song	ins/song	cor/song	chords/song	time/song	tot. time
`small`	0.83	2.79	3.63	42.49	10.00 ms	0.72 s
`large`	3.38	9.85	13.24	62.05	76.53 ms	384.81 s

Table 1: The deleted, inserted, and total number of corrections per song; the total number of chords per song; the parsing time per song; and the total parsing time in seconds.

with modulations, atonal harmonies, and a variety of errors. Still, HarmTrace keeps the number of deleted chords under six percent of the total chords parsed.

When we compare the parsing results of the `small` dataset with the results of De Haas et al. (2009), we notice that HarmTrace is much faster. The Java-based parser used by De Haas et al. takes more than 9 min to parse the dataset. We cannot compare the parsing results of the `large` dataset because the majority of the pieces is rejected by their grammar. This emphasises how important the error-correction process is.

6 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 fast, and is robust against noisy data. However, HarmTrace currently features only parallel key distinction, and no support for full modulation. Although the model presented has a bias towards jazz harmony, we have furthermore shown that it can be used to analyse 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 12-bar blues, whereas our model aims to formalise the core of tonal harmony with a bias towards jazz harmony, including the 12-bar blues. Although our work draws on the work of Rohrmeier (2007, 2011), there are also considerable contrasts. The most pronounced difference to Rohrmeier's CFG is that the latter features modulation/tonicisation. By tonicising to a local tonic, chords are analysed with respect to that local tonic. As a consequence, his approach can explain secondary dominants by tonicisation, while the HarmTrace model uses a more jazz-oriented approach to harmony by identifying ii-V-I motions, e.g. 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 local subdominant/dominant pair. A specification for this would be easy to add to the HarmTrace model. However, implementing both tonicisations and secondary dominants, as Rohrmeier suggests, will be problematic since both rules explain the same phenomena. After all, the preparation F by C can be explained both by the tonicisation rules as well as by the secondary dominant rules. This will 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 a model that is implementable allows it to be evaluated empirically and used in practice. Hence, we argue that if a model designer wants their model to have practical value, they should keep in mind how the model can be implemented. As we have seen in this paper, 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 modelling of harmonic analysis, and that it will prove itself useful in tasks such as harmonic similarity estimation and chord labelling.

7 Acknowledgements

This work has been partially funded by the Dutch ICES/KIS III Bsik project MultimediaN, and by the Portuguese Foundation for Science and Technology (FCT) via the SFRH/BD/35999/2007 grant. We thank Martin Rohrmeier for the inspiring discussions and ideas about modeling harmony, and Anja Volk for comments on a draft version of this paper.

References

Bird, R. 1998. *Introduction to Functional Programming using Haskell*. Prentice Hall Press.

Chemillier, M. 2004. "Toward a formal study of jazz chord sequences generated by Steedman's grammar." *Soft Computing-A Fusion of Foundations, Methodologies and Applications* 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*. 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.

Gannon, P. 1990. "Band-in-a-Box." PG Music. URL `http://www.pgmusic.com/`.

De Haas, W. B., J. P. Magalhães, F. Wiering, and R. C. Veltkamp. 2011. "HarmTrace: A similarity framework for tonal harmony based on functional harmony analysis." In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*.

De Haas, W. B., M. Robine, P. Hanna, R. C. Veltkamp, and F. Wiering. 2010. "Comparing harmonic similarity measures." In *Proceedings of the 7th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. pp. 299–315.

De Haas, W. B., M. Rohrmeier, R. C. Veltkamp, and F. Wiering. 2009. "Modeling harmonic similarity using a generative grammar of tonal harmony." In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*. pp. 549–554.

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., M. Sandler, S. Abdallah, and E. Gómez. 2005. "Symbolic representation of musical chords: A proposed syntax for text annotations." In *Proceedings of the 6th International Symposium on Music Information Retrieval (ISMIR)*. pp. 66–71.

Jeuring, J., S. Leather, J. P. Magalhães, and A. Rodriguez Yakushev. 2009. "Libraries for generic programming in Haskell." In P. Koopman, R. Plasmeijer, and D. Swierstra, (editors) *Advanced Functional Programming, 6th International School, AFP 2008, Revised Lectures*, *Lecture Notes in Computer Science*, volume 5832. Springer, pp. 165–229.

Lerdahl, F., and R. Jackendoff. 1996. *A Generative Theory of Tonal Music*. MIT Press.

Magalhães, J. P., and W. B. de Haas. 2011. "Functional Modelling of Musical Harmony—an Experience Report." In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming*.

Marsden, A. 2010. "Schenkerian analysis by computer: A proof of concept." *Journal of New Music Research* 39(3):269–289.

Mauch, M., S. Dixon, C. Harte, M. Casey, and B. Fields. 2007. "Discovering chord idioms through Beatles and real book songs." In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*. pp. 255–258.

Pachet, F. 1999. "Surprising harmonies." *International Journal of Computing Anticipatory Systems* 4:139–161.

Piston, W., and M. DeVoto. 1991. *Harmony*. Victor Gollancz.

Rameau, J.-P. 1971. *Treatise on harmony*. Dover Publications. Translated by Philip Gossett.

Riemann, H. 1893. *Vereinfachte Harmonielehre; oder, die Lehre von den tonalen Funktionen der Akkorde*. Augener.

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 A. Georgaki, Kouroupetroglou, (editor) *Proceedings of the 4th 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.

Schenker, H. 1935. *Der Freie Satz. Neue musikalische Theorien und Phantasien*. Universal-Edition, Vienna.

Schrijvers, T., S. Peyton Jones, M. Sulzmann, and D. Vytiniotis. 2009. "Complete and decidable type inference for GADTs." In *Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming*. pp. 341–352.

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*, chapter 15. Psychology Press, pp. 305–318.

Swierstra, S. D. 2009. *Combinator Parsing: A Short Tutorial*. Springer-Verlag, pp. 252–300.

Temperley, D. 2001. *The cognition of basic musical structures*. Cambridge, MA, MIT Press.

Temperley, D., and D. Sleator. 1999. "Modeling meter and harmony: A preference-rule approach." *Computer Music Journal* 23(1):10–27.

Various authors. 2005. *The Real Book*. Hal Leonard Corporation, 6th edition.

Winograd, T. 1968. "Linguistics and the computer analysis of tonal harmony." *Journal of Music Theory* 12(1):2–49.