MODELING HARMONIC SIMILARITY USING A GENERATIVE GRAMMAR OF TONAL HARMONY

W. Bas de Haas
Utrecht University
Bas.deHaas@cs.uu.nl

Martin Rohrmeier
University of Cambridge
mr397@cam.ac.uk

Remco C. Veltkamp
Utrecht University
Remco.Veltkamp@cs.uu.nl

Frans Wiering
Utrecht University
Frans.Wiering@cs.uu.nl

ABSTRACT

In this paper we investigate a new approach to the similarity of tonal harmony. We create a fully functional remodeling of an earlier version of Rohrmeier's grammar of harmony. With this grammar an automatic harmonic analysis of a sequence of symbolic chord labels is obtained in the form of a parse tree. The harmonic similarity is determined by finding and examining the largest labeled common embeddable subtree (LLCES) of two parse trees. For the calculation of the LLCES a new $O(\min(n,m)nm)$ time algorithm is presented, where n and m are the sizes of the trees. For the analysis of the LLCES we propose six distance measures that exploit several structural characteristics of the Combined LLCES. We demonstrate in a retrieval experiment that at least one of these new methods significantly outperforms a baseline string matching approach and thereby show that using additional musical knowledge from music cognitive and music theoretic models actually helps improving retrieval performance.

1. INTRODUCTION

Harmonic Similarity is a relatively new research topic within Music Information Retrieval (MIR) that is concerned with determining the similarity of the chord sequences in songs and enables users to search for songs on the basis of their harmony. Retrieval based on harmony offers obvious benefits: it allows for finding cover songs (especially when melodies vary), songs of a certain family (like Blues or Rhythm Changes), or variations over standard basses in instrumental baroque music, to name a few. So far, very few measures of harmonic similarity have been proposed. De Haas et al. [1] developed a distance measure based on Lerdahl's Tonal Pitch Space [2].

When researching MIR, it is important to realize that only part of the information needed for good similarity judgment can be found in the musical data. Musically schooled as well as unschooled listeners have extensive knowledge about music [3,4] and one important task of a MIR researcher is to select or develop the appropriate music cognitive and music theoretical models that provide the knowledge needed for making good similarity judgments. We strongly believe that such a model is necessary, and that systems without such additional musical knowledge are incapable of capturing a large number of important musical features. In this study we report a new method of harmonic similarity matching that applies a remodeling of Rohrmeier's [5] phrase-structure grammar for tonal harmony as underlying cognitive and music theoretical model.

In analogy to linguistics, various hierarchical models of musical structure have been proposed since the 1980s and have been brought up recently in cognitive and computational discussions [6–8]. In this context, Rohrmeier's generative grammar of diatonic harmony [5] transfers notions about the hierarchical organization of tonal music [6,7] to the area of harmony. It is based on the assumption that, within a sequence of harmonies, different chords have different degrees of stability and dependency, based on their position within the hierarchical structure. In a chord sequence several chords may be replaced, inserted or omitted in such a way that the harmonic structure remains intact, whereas the changes of structurally important anchor chords may result in large structural modifications of the entire dependency structure of the harmony sequence. These dependency features and relationships resemble constituent structure and dependencies in linguistic syntax and can be modeled with a grammatical formalism [9].

These variable relationships between chords and their structural roles motivate the rationale not to base our harmony matching methods on sequence matching methods–which assume the equal importance of all chords in a sequence–but on a hierarchical formalization that incorporates the differences in structural function. Figure 1, displaying two versions of the jazz standard *Take the 'A' train*, illustrates this idea. Even though both sequences appear to be substantially different when compared element by element, an analysis of their formal dependencies reveals that both derive from a common harmonic pattern that is represented by the parse trees and fits human intuition.

We present a fully functional remodeling of Rohrmeier's grammar [5], which parses sequences of symbolic chord labels and returns parse trees like the ones in Figure 1, in section 3. A parse tree is more than a harmonic analysis alone, since it contains all the structural relations of the harmonies used in a song, and is therefore very suitable for determining harmonic similarity. We compare parse trees by finding and examining the combined Largest Labeled

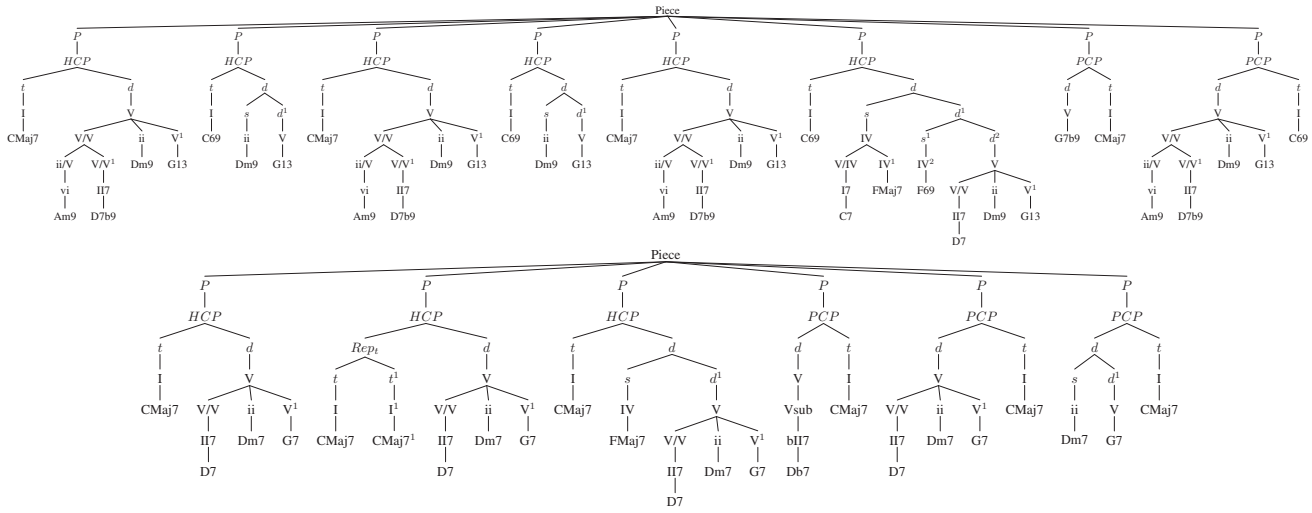

Figure 1. Two parse trees of different versions of the same jazz standard *Take the 'A' train*. The leafs of the tree represent the actual chords of the sequence.

Common Embeddable Subtree (LLCES). The LLCES is the tree that can be included in both parse trees while maintaining the labels and the ancestor relations. In section 4.2 we present a new algorithm that finds the LLCES. Using the LLCES we define a series of similarity measures for tonal harmony.

Contribution: First, we present a remodeling of a formal grammar for tonal harmony and propose solutions for some of the typical problems of its application. Second, we present a new $O(\min(n, m)nm)$ time algorithm that calculates the LLCES, where n and m are the sizes of the trees. Third, six LLCES based distance measures are defined to compare the parse trees. Last, the retrieval performance of these distance measures is experimentally verified on a dataset of 72 symbolic chord sequences of jazz standards.

2. RELATED WORK

In the last century, numerous formal theoretical approaches to western tonal music have been proposed. Formalizing Schenker's theory [6], Lerdahl and Jackendoff [7] proposed a generative theory that organized western tonal music by recursive hierarchical dependency relationships between musical elements in terms of time-span reduction and prolongation structure. They formalized the interaction between these structures with metrical and grouping structure in terms of constraint based preference rules. Similarly, there is some theoretical evidence that tonal harmony is organized in a comparable, hierarchical way. Early attempts by Kostka and Payne ([10] ch. 13) and Baroni [11] suggest that harmony is organized in hierarchical layers. Current theoretical approaches [5,12–15] suggest that the structure of harmony sequences exceeds the simplicity of a straightforward chord transition table (or finite-state grammar [9]), like Piston's table of root progressions [16], and may be modeled by hierarchical, context-free or phrase-structure grammars [9]. Pachet [17] proposes a set of rewrite rules for jazz harmony similar to Steedman's grammar [12]. He shows that these rules could be learned form chord sequence data in an automated fashion. Rohrmeier [5] gave an encompassing account how tonal harmonic relationships may be formalized using a generative context-free grammar with variable binding.

3. A GRAMMAR FOR TONAL HARMONY

The generative formalism proposed by Rohrmeier [5] expands on earlier approaches in a number of ways. Steedman's approach [12,13] is merely concerned with Blues progressions and, featuring only seven context-sensitive rules (with variations), omits a number of theoretically important features to extend to a broader domain. Rohrmeier's approach extends on these ideas and gives an overarching account of tonal harmony and tonal-phrase structure independently of a specific style or musical form. In addition, it proposes to incorporate the structural distinctions between form, theoretical harmonic function [18], scale degree prolongation [6,7] and surface feature realization into different levels of the syntactic derivation. The present study proposes a remodeling of the grammar without modulation and with limited local tonicization and scale adaptation in order to reduce the complexity for the implementation of a first-stage working system. The current remodeling was optimized for jazz, but the aim is to develop a set of core rules that explain basic harmony structures which can be augmented with style specific rules.

The grammar incorporates four levels: a phrase level, functional level, scale-degree level and surface level. The phrase level divides a piece into phrases, the functional level specifies the functional role a certain scale-degree has within a phrase. The scale-degree captures the relation between the chord and the key and the surface level expresses the actual chord with all its possible additions, inversions, etc.

Below, the main rules of the grammar are listed in order to give an outline of the architecture of the grammar. A piece always consists of one or more phrases (P). On this phrase level the grammar distinguishes two types of phrases: phrases which end on a perfect cadence (PCP) and phrases which end with a half-cadence (HCP). Per-

fect cadence phrases are distinguished by ending with a tonic function (t) upon which all subordinate harmonic elements are dependent, whereas half-cadence phrases force a phrase to end with a dominant function (d) which results in a tonicization of, or a perfect or imperfect cadence on the dominant.

1. $Piece \rightarrow P_+$
2. $P \rightarrow PCP$
3. $P \rightarrow HCP$
4. $PCP \rightarrow d\ t_+ \mid d\ d\ t_+ \mid t\ d\ t$
5. $HCP \rightarrow t_+\ d$

At the functional level, the grammar encapsulates core relationships between the three main harmonic functions: tonic (t), dominant (d) and subdominant (s).

6. $d \rightarrow s\ d$
7. $t \rightarrow tpg$

These functional rules can be applied recursively, but finally translate into scale-degrees. Rule 9 deals with certain forms of parallels (tpg).

8. $t \rightarrow$ I
9. $tpg \rightarrow$ vi | iii
10. $d \rightarrow$ V | vii^0
11. $s \rightarrow$ ii | IV

The functional level also incorporates a number of additional prolongational rules that allow for the derivation of more distant harmonic structures such as the preparatory use of iii and tritone substitutions. Rule 12 incorporates a feature specifically added for modeling the prototypical II-V-I sequences in jazz harmony that are less frequent in other styles.

12. $x \rightarrow$ V(x) x | ii(x) V(x) x for any scale degree x
13. IV $\rightarrow$ iii IV
14. V(x) $\rightarrow$ bII(x) for any scale degree x

At the surface level scale degree symbols are translated into the actual surface chord. These translation steps are straightforward when the key is known beforehand. For instance, a VI symbol in the key of C minor would translate into an A♭-chord. In addition, elaborations of chords are added at this level of description: a surface realization of a VI chord may result in a A♭6 chord. Some of these surface elaborations of chords are tied to their structural functions (strong typing), e.g. an Em7♭5 chord label indicates a subdominant function ii in a ii-V-I sequence, or a D7 chord label indicates a dominant function (except in blues contexts where minor sevenths loose their functional connotation).

3.1 Implementation and Parsing

There are some additional rules that have been implemented, but are not described here. Among these are rules for typical voice-leading and prolongational structures and some typical borrowings from the parallel key. Since we have not incorporated modulation yet, it is necessary to label these phenomena to be able to explain the remainder of the piece. Furthermore, there are rules that deal with typical well-known diminished chord transitions in various descending and ascending forms.

The grammar as specified above is not strictly a context free grammar, because rule 12 and rule 14 use a variable binding. However, by expanding a rule for every element x that it holds, a set of context free rules can be created that yields the exact same outcome. Having a context free grammar, a free Java implementation [19] of an Earley Parser [20] is used to parse the chord sequences in $O(n^3)$ time, where n is the number of chords.

Context free grammars often create multiple ambiguous parse trees. To select the optimal parse tree out of the set of parse trees, we provided the rules with weights (set by hand) and defined the total weight of a parse tree as the product of the weights of the rules used in its construction. Because of this, some rules have less chance to be used in the final derivation. This allows to select the best tree from the ambiguous parse trees. The complete grammar as well as the lead-sheets of the examples in Fig 1 are available online [1] .

4. COMMON HIERARCHICAL STRUCTURES

In this section we present six distance measures for the parse trees generated by the grammar discussed in the previous section. For the comparison of parse trees, we propose an approach based on the problem of tree inclusion, which is elaborately dealt with in [21]. Given the parse trees of two songs, the general idea is to find the collection of largest labeled common embeddable subtrees (LLCESs) for every combination of phrases. The LLCES is the largest tree that is included in both parse trees. This means that there exists a one-to-one mapping from the nodes of the LLCES to the nodes with the same label in both parse trees that preserves ancestor relations, but not necessarily parent relations. When processing harmony parse trees, this is a natural thing to do because typically a chord progression is augmented by adding a structure to the left branch and repeating the right branch, e.g. when a Dm is prepared by an A7 chord. Hence, if both trees are similar, the LLCES reflect the structure of the parse trees it is generated from, and if both trees are dissimilar, the resulting LLCES will be much smaller and less grammatical. In the next sections we explain the calculation of the LLCES, and how we use it to define six distance measures.

4.1 Preliminaries

A rooted tree is a structure denoted with $P = (V, E, P_{root})$, where V is a finite set of nodes, E is the set of edges con-

[1] http://give-lab.cs.uu.nl/music/

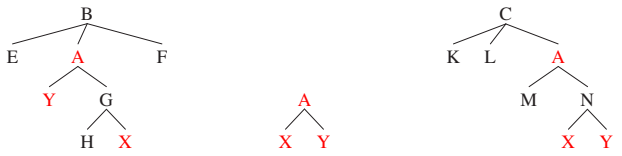

Figure 2. An example of a rooted by A that can be embedded into two larger trees rooted by B and C.

necting nodes in V, and P_{root} is the root of the tree P. The nodes of the parse trees generated by the grammar of section 3 are all labeled and the label of node v is denoted with $label(v)$. The subtree of P rooted at node v is denoted with $P[v]$ and $children(v)$ denotes the subset of V with nodes that have v as parent. Similarly $desc(v)$, denotes the decedents of v, i.e. the subset of V that is contained in $P[v]$ and have v as ancestor. Furthermore we use a few additional functions, $po(v)$ denotes the post order number that is assigned to a node v in a postorder traversal. $depth(P)$ denotes the depth of a tree, i.e. the number of nodes in the longest path from leaf to the root. Finally, the $degree(P)$ is the degree of a tree, i.e. the maximum number of children.

We say that a tree $P = (V, E, P_{root})$ is *included* in a tree $T = (W, F, T_{root})$ if there exists an embedding of P into T. An embedding is an injective function f, mapping each node in P to a node in T, that preserves labels and ancestorship. Figure 2 shows an example of an included tree. Note that a left-to-right ordering of the descendants is not required. Formally, this means that for all nodes u and v in P it is required that:

1. $f(u) = f(v)$ if and only if $u = v$,
2. $label(u) = label(f(v))$,
3. u is an ancestor of v in P if and only if $f(u)$ is an ancestor of $f(v)$ in T.

4.2 Largest Labeled Common Embeddable Subtree

We are not aware of an algorithm that calculates the largest common embeddable subtree for labeled trees. Gupta and Nishimura [22] have developed a $O(n^{2.5} \log n)$ time algorithm for finding this tree for two unlabeled trees. The algorithm we present here calculates the largest common embeddable subtree for the labeled case and expands on the general tree matching ideas as described in [21], ch. 3.

Algorithm 1 calculates the LLCES of two trees $P = (V, E, P_{root})$ and $T = (W, F, T_{root})$. To store the nodes of the subtrees of the LLCES the algorithm uses a table M such that $M[po(w)]$ stores the subtrees that can be embedded into both P and $T[w]$. The algorithm builds the LLCES up from the leaves by traversing the nodes of T and P in postorder. When a node v with an identical label as w is encountered (line 5), the algorithm creates a new node x with the same label as v. In case w is a leaf, x is stored in M (lines 8–9). In case w is an internal node, we look up the subtrees in M that match the children of w. Because the tree is processed in postorder these nodes were previously stored in M and can be retrieved from $M[po(w')]$ for each child w'. If a previously stored subtree rooted by

Algorithm 1 Largest Labeled Common Embeddable Subtree

```
1:  procedure LLCES(P, T)
2:      M ← ∅
3:      for all w ∈ W in postorder do
4:          for all v ∈ V in postorder do
5:              if label(v) = label(w) then
6:                  x ← new node
7:                  label(x) ← label(v)
8:                  if children(w) = ∅ then
9:                      add x to M[po(w)]
10:                 else
11:                     for all w' ∈ children(w) do
12:                         for all x' ∈ M[po(w')] do
13:                             if x' ∈ desc(v) then
14:                                 add (x, x') to M[po(w)]
15:                             else
16:                                 add x' to M[po(w)]
17:                             end if
18:                         end for
19:                     end for
20:                     add x to M[po(w)]
21:                 end if
22:             end if
23:         end for
24:         if M[po(w)] = ∅ then
25:             for all w' ∈ children(w) do
26:                 add M[po(w')] to M[po(w)]
27:             end for
28:         end if
29:     end for
30:     return M[po(T_root)]
31: end procedure
```

x' is a descendant of v, this subtree becomes a child of the new node x, by adding a new edge (x, x') to $M[po(w)]$ (lines 10–15). Otherwise, if x' is not a descendant of v, x' is stored in $M[po(w)]$ (line 16). After all, a common ancestor can show up in a next iteration. Finally, the new subtree x is stored in M as well (line 20). If the label of w does not match any of the labels of the nodes in P, the subtrees stored in M for all children w' of w are added to $M[po(w)]$ (lines 24–28). This process continues until all nodes of T have been matched against all nodes of P and finally $M[po(T_{root})]$, the LLCES of P and T, is returned. A drawback of our algorithm is that it is incapable of dealing with duplicate labels. Therefore we number the duplicate labels that descent the same phrase.

The running time of the algorithm is dominated by the lines 3-23. For each of the $O(nm)$ combinations of w and v (lines 3, 4) a constant time test is performed. Because the labels are unique, only $\min(n, m)$ times each of the $O(n)$ nodes in the subtrees that has been stored in $M[po(w)]$ so far (line 12), is checked against each of the $O(m)$ descendants of node v (line 13). This results in a time complexity for the whole algorithm of $O(\min(n, m)nm)$.

4.3 Distance Measures

We base the distance measures on the LLCES, but we do not calculate the LLCES of two parse trees directly for two reasons. First, as we can see in Figure 1, there are quite some duplicate labels in the parse trees which our algorithm cannot handle. Second, if a parse tree of a song contains a repetition of a phrase that the matched song does not have, the repeated phrase cannot be matched. To solve

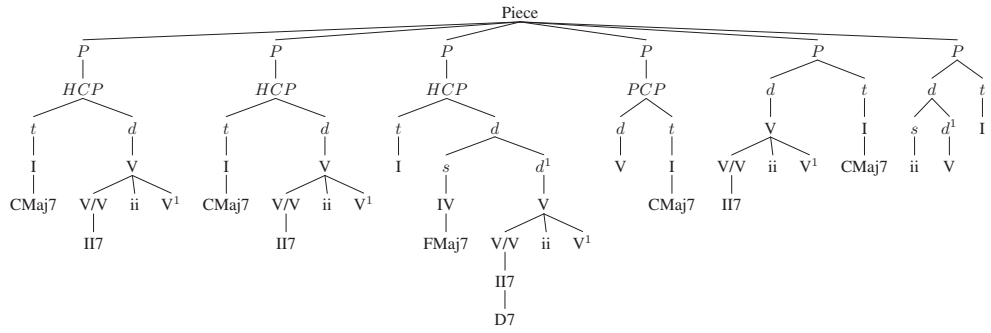

Figure 3. The Combined LLCES for every combination of phrases of the parse trees of *Take the 'A' Train* as in Fig. 1.

these two problems we compare parse trees in a phrase-wise fashion. For every phrase in the target tree T we calculate the LLCES for every phrase of the pattern parse tree P and pick the largest LLCES to create a *Combined LLCES* (see Fig. 3). The duplicate labels are re-labeled per phrase too in preorder (see the superscripts in Fig. 1 and 3). Because the number of duplicate labels per phrase is small, the labellings will be nearly identical if two trees have a similar harmonic structure. Note that if the two parse trees T and P have a different number of phrases, the structure of the Combined LLCES will differ if P is used as a target tree, because for every phase in the T a LLCES is constructed. This makes every Combined LLCES based measure is asymmetrical.

We propose three distance measures for sequences of symbolic chord labels based on the Combined LLCES:

1. *Relative Combined LLCES Size Distance (Rel)*: By dividing the number of nodes in the target tree T by the number of nodes in the Combined LLCES a distance measure between 0 and 1 is obtained that is normalized by the size of T.
2. *Grammar Violation Distance (Viol)*: if two trees are not similar, the combined LLCES will contain connections between nodes that cannot be explained by the grammar. By dividing the number of nodes in the target tree T (which are grammatical by definition) by the number of grammatical nodes in the Combined LLCES we obtain a distance measure between 1 and 0 that is normalized by the size of T.
3. *Average Depth Distance (Dep)*: if trees are very similar, the level of complexity in the harmonic structure in the Combined LLCES will be comparable to the level of complexity target tree T. By dividing the average leaf depth of T by the average leaf depth of the Combined LLCES, we obtain a distance measure between 1 and 0, that is normalized by the size of T.

One can observe in Figure 1 that, having the actual parse tree structure, the actual chord labels are not of much importance anymore. Given two similar sequences, it is rather arbitrary whether the chords labels match or not: the structure of the harmony determines the similarity. Therefore we can remove each leaf node describing a surface chord from the Combined LLCES and target trees. The structure of the phrase, functional and scale-degree level remains unchanged. As a consequence, this yields three additional harmonic distance measures that are concerned with the structure of the harmony only. Other Combined LLCES distance measures can be thought of.

5. EXPERIMENT

We have evaluated the six LLCES based distance measures described in the previous section in an experiment. We assembled a dataset of 72 symbolic chord label sequences extracted from user-generated Band-in-a-Box files that were collected on the Internet. Band-in-a-Box is a software package that generates accompaniment given a certain chord sequence provided by the user. This dataset consists of 51 different pieces of which 16 pieces contain two or three versions, forming 16 song classes. These pieces are all jazz standards from before the 1970's and can all be found in the Real Book [23] or similar sources. All parse trees of these pieces are available online [2]. The task is to retrieve the other versions of a song class, given a certain query song from that class. All songs containing more than one version are used as a query and the rankings are analyzed by calculating the mean average precision (MAP). To place the results in perspective, we calculate the edit distance [24] between all chord sequences, represented as a string of chord labels, as a baseline measure.

The results are presented in Table 1. It seems that all Combined LLCES based methods perform better than the baseline edit distance, but only the difference between the *Viol* distance measure without chord symbol nodes scores significantly better than the baseline edit distance ($p < .01$, two-tailed T-test). The results show therefore that the number of grammatical connections in the Combined LLCES is a good indicator for harmonic similarity. The lack of significance of the other measures might be explained by the limited size of the relatively small dataset. However, the experiment does show that a matching method that analyzes the structure of the harmony outperforms a sequence-based method that does not use any musical knowledge.

6. CONCLUDING REMARKS

This paper introduced a new approach to harmonic similarity. We showed that a grammar of tonal harmony can

[2] http://give-lab.cs.uu.nl/music/

	Chord Symbols			*No Chord Symbols*			
Distance:	Rel	Viol	Dep	Rel	Viol	Dep	Edit
MAP:	0,79	0,81	0,72	0,81	0,86	0,73	0.67

Table 1. The MAP of the six Combined LLCES based similarity measures and a baseline edit distance.

be adapted in such a way that is usable for matching harmony sequences. However, there are some open issues. At the moment we cannot calculate distance measures to pieces that do not parse and for every grammar there are always pieces imaginable that do not parse. A solution to this problem can be found in partial matching. Often only one or two chords cannot be explained by the grammar. By removing these chords and parse the left and the right side separately, it is possible to obtain a parse tree that can be used for matching.

A property of context free grammars is that sequences can have multiple ambiguous parse trees. Using the grammar presented here, many chord sequences are intrinsically ambiguous and have multiple derivations. One solution might be to incorporate intrinsically ambiguous parse trees in the creation of the Combined LLCES. Nevertheless, it is important to keep the number of unwanted ambiguous parse trees as low as possible. By making the grammar strongly typed and adding weights to rules, we controlled the number and the selection of parse trees. Still, the grammar as presented here features several problems with respect to the parsing of phrase boundaries, which constitutes a main source of ambiguities (as in Fig. 1). A set of additional preference rules will be designed for future versions of the model to rule out unlikely phrase-boundaries. These will be based on metrical information which is not yet incorporated in the present model. Yet another way of improving the expressive power of the grammar and limiting the number of ambiguous parse trees at the same time, is to start parsing with a very strict grammar and, only after a rejection of the chord sequence, to add more loosely typed rules that can explain the more exotic harmonic phenomena.

The research presented here demonstrates how a grammar of harmony may characterize harmonic similarity in a musical way. This will have a large impact on the quality of the representation, analysis and retrieval of tonal music. This research also provides a case study that demonstrates the importance of cognitive and theoretic models of music in the design of appropriate methods for MIR tasks that have been neglected so far because of their inherent musical complexity.

7. ACKNOWLEDGMENTS

This work was supported by the Dutch ICES/KIS III Bsik project MultimediaN and in part by Microsoft Research through the European PhD Scholarship Programme.

8. REFERENCES

[1] W.B. de Haas, R.C. Veltkamp, and F. Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 51–56, 2008.

[2] F. Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2001.

[3] I. Deliège, M. Mélen, D. Stammers, and I. Cross. Musical Schemata in Real Time Listening to a Piece of Music. *Music Perception*, 14(2):117–160, 1996.

[4] E. Bigand. More About the Musical Expertise of Musically Untrained Listeners. *Annals of the New York Academy of Sciences*, 999:304–312, 2003.

[5] M. Rohrmeier. A Generative Grammar Approach to Diatonic Harmonic Structure. In Anagnostopoulou Georgaki, Kouroupetroglou, editor, *Proceedings of the 4th Sound and Music Computing Conference*, pages 97–100, 2007.

[6] H. Schenker. Der Freie Satz. Neue musikalische Theorien und Phantasien, 1935.

[7] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1983.

[8] A.D. Patel. Language, Music, Syntax and the Brain. *Nature Neuroscience*, 6:674–681, 2003.

[9] N. Chomsky. *Syntactic Structures*. Mouton, 1957.

[10] S. Kostka and D. Payne. *Tonal Harmony with an Introduction to 20th-century Music*. McGraw-Hill, 1984.

[11] M. Baroni, S. Maguire, and W. Drabkin. The Concept of Musical Grammar. *Music Analysis*, 2(2):175–208, 1983.

[12] M. J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 2(1):52–77, 1984.

[13] M. J. Steedman. *The Blues and the Abstract truth: Music and Mental Models*, chapter 15, pages 305 – 318. Psychology Press, 1996.

[14] M. Chemillier. Toward a Formal Study of Jazz Chord Sequences Generated by Steedmans Grammar. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):617–622, 2004.

[15] S. Tojo, Y. Oka, and M. Nishida. Analysis of Chord Progression by HPSG. In *Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pages 305–310. ACTA Press Anaheim, CA, USA, 2006.

[16] W. Piston. *Harmony*. Norton, W. W. & Company, New York, 1948.

[17] F. Pachet. Surprising Harmonies. *International Journal of Computing Anticipatory Systems*, 4, 1999.

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

[19] S. Martin. Pep is an Earley Parser. http://www.ling.ohio-state.edu/~scott/, 2007.

[20] J. Earley. An Efficient Context-free Parsing Algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[21] P. Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, Departement of Computer Science, University of Helsinki, November 1992.

[22] A. Gupta and N. Nishimura. Finding Largest Subtrees and Smallest Supertrees. *Algorithmica*, 21(2):183–210, 1998.

[23] Various Authors. *The Real Book*. Hal Leonard Corporation, 6th edition, 2004.

[24] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.