Automating Functional Harmonic Analysis: The Funchal System

Ricardo Scholz
reps@cin.ufpe.br

Vítor Dantas
vcd2@cin.ufpe.br

Geber Ramalho
glr@cin.ufpe.br

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.732-970 – Recife – PE – Brazil

Abstract

Functional harmonic analysis is an important task in music composition, accompaniment, arrangement and others. However, the solutions are still not satisfactory. The proposed process is divided into two levels: the former extends one of previous works in the domain to carry out a richer analysis of chord grids and is where the very analysis is performed, and the latter is devoted to correct some conceptual inconsistencies concerning enharmonic spelling of chords. Both levels use an engine to make inferences on some rule bases, which can be easily improved by addition of new rules.
Keywords: functional harmonic analysis, roman numeral harmonic analysis.

1. Introduction

Functional harmonic analysis consists of attributing functions to each chord of a given chord grid, considering the context in which the chords are inserted, i.e., the preceding and the following chords. This analysis provides the tonality of each excerpt of the grid as well as the role of each chord.

Many tasks in music, such as improvisation, re-harmonization, arrangement, music composition and accompaniment, require functional harmonic analysis as a preliminary step. The main targets criteria for evaluating a system that performs functional harmonic analysis are correctness, extensibility and ability to recognize most of the patterns used. A user interface that allows edition of the results of the analysis and accepts as input a simple format, such as text or songs from the web, would also be a good feature.

The previous works studied did not reach some of these criteria. They do not cover many of the patterns found in functional music. Moreover, most of them are not easily extensible and use specific input data formats.

Our proposed process is divided into two levels: analysis and chord names correction. The former extends the work of Pachet [8], reused by Ramalho [10]. It comprises three steps: chord patterns matching, overlapping removal and gaps classification. We extended the approach to deal with more complicated harmony structures, such as modal borrowing and secondary dominants. The latter level, chord names correction, was added to the approach in order to make the former more robust and able to process thousands of chord grids available on the Web, which often contain mistakes concerning the enharmonic spelling of chords.

Next section describes the functional analysis task. Section 3 will present the state of the art on this subject. Next, section 4 discusses in depth the approach that we used in our system to do the automatic functional harmonic analysis. In section 5, we show the results obtained. Last section presents the conclusions of this work.

2. The Task

We will assume that readers already have background knowledge on some concepts in music, as enharmonic notes, harmonic field, function of a chord and modal borrowing. The chords notation will follow the Brazilian pattern used in Chediak's songbooks [11], which can be found in web stores such as Amazon, but dissonances will be noted in superscript.

Notes whose pitches are the same and names are different (e.g., D# and Eb), are called enharmonic notes. An harmonic field is the set of the chords composed by the notes of a given scale (e. g. the seven chords that compose the C major harmonic field are C^{7M}, Dm^{7}, Em^{7}, F^{7M}, G^{7}, Am^{7} and $Bm^{(b5)7}$).

In a given harmonic field, each chord has a function relative to the harmonic field's root (e. g., in C major they are I^{7M}, IIm^{7}, $IIIm^{7}$, IV^{7M}, V^{7}, VIm^{7} and $VIIm^{(b5)7}$, each one associated with its respective chord in C

major harmonic field, meaning I^{7M} associated with C^{7M}, IIm^7 with Dm^7, and so on). Modal borrowing occurs when a tonality (e. g. C major) "borrows" a chord from the harmonic field of the same tonality, but in a different mode (e.g., C minor).

Functional analysis is a difficult task, since there is often more than one function for a chord. In fact, we are dealing with a language whose grammar is, by definition, context-dependent. For example Ab^{7M} could have many different functions, as I^{7M} in Ab major, bVI^{7M} in C minor, IV^{7M} in Eb major, III^{7M} in F minor and so on. However, given the context shown in Figure 1, the most plausible analysis for Ab^{7M} is bVI^{7M}. The context considered may be larger than one chord long in each direction.

bVI^{7M}	V^7	I^{7M}
Ab^{7M}	G^7	C^{7M}

Figure 1. Context-dependent analysis.

In addition to the problem of context-dependence, we have one more difficulty: harmonic analysis is an activity that demands a great amount of specialized knowledge. Without formal rules to carry through the analysis, the modeling of this knowledge, that indeed allows ambiguous situations, seems to be even more complex.

An example of ambiguous situation can be found in trying to identify a deceptive cadence as "V^7 VIm^7" in a major tonality and a cadency like "$bVII^7$ Im" in a minor tonality. It is impossible to define whether a sequence as "G^7 Am" has the functions "V^7 VIm^7" in C major, or "$bVII^7$ Im" in A minor.

However, observing the context in which the sequence is inserted, the classification is possible. If the whole sequence is "F^{7M} Em^7 Am^7 Dm^7 G^7 Am", it is clear that the harmonic field is C major, and we are dealing with a deceptive cadency, so the functions would be, respectively, "IV^{7M} $IIIm^7$ VIm^7 IIm^7 V^7 VIm". However, if the sequence is "Dm^7 $Bm^{7(b5)}$ G^7 Am" it is possible to infer that its harmonic field is A minor and the chords functions are "IVm^7 $IIm^{7(b5)}$ $bVII^7$ Im". The chosen method leaves the Am as a gap and classifies it based on the previous pattern's tonality, as shown in Figure 2.

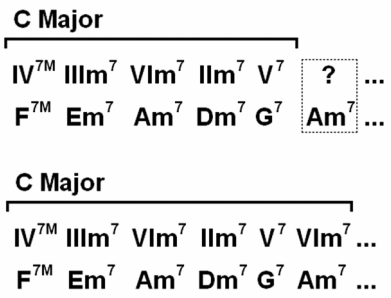

Figure 2. Ambiguous chord classification.

3. State of the Art

The perception of harmony in music is a task that is done very naturally by humans, but has been proven to be a complex task to describe algorithmically. Many authors have done studies in psychology to understand the human perception of stability and similarity relations between chords in a piece, and have tried to use spatial models to represent these relations [5].

In fact, a good way to gain insight on psychological processes is to approach them from a computational point of view. For instance, Bharucha have done experiments with connectionist networks to simulate harmonic perceptions [2]. However, some works assumed basic harmonic structures as part of the input, in form of rule-bases, such as Lerdahl and Jackendoff's generative theory of tonal music [6] and Narmour's implication/realization model [7].

Temperley did recently a good extention of these works, with an approach using spacial representations and rules [17]. He also has the merit for actually implementing his solutions and for doing extensive tests. Another thing to mention is that he ignores pitch spelling in his rules, assuming that the spelling is not relevant to the perception and analysis of harmony [16]. To avoid the flaws of assuming wrong predefined structures, some authors have tried to use case-based reasoning, relaying on the fact that the harmony is in the music and not in the rules [14].

Besides, a number of studies deal with what is known as roman numeral harmonic analysis, that is the way it is taught in music schools. However, most of these studies use this task as an intermediate step, aiming for other ends such as accompaniment [10] and composition [18], so that the solutions are often very specific to each problem.

Some authors, approaching roman numeral harmonic analysis, proposed context-free grammars to realize the analysis iteratively, generating more complex harmonic structures on each iteration [18] [15]. Pachet's approach to this problem is different and particularly interesting. It consists in dividing the

analysis in three steps, according to three defined rule-bases: first, there is a search for recurrent patterns (most from Jazz harmony), then overlapping patterns are resolved and finally the yet unclassified chords are given a function. [8].

In our work, we used Pachet's division in levels, and we have made a number of extensions to recognize more complex harmonic structures such as modal borrowing, secondary dominant and composite functions. Our system accepts chord grids as input, in the way they are commonly found in the web, to avoid a solution too specific. We also added new a step to correct the names of chords that can be misspelled in the input. "Behind the scenes" our analysis level deals only with pitch information, without considering the spelling, such as Temperley's analysis did [17].

4. Automatic Functional Harmonic Analysis

The approach proposed works like growing islands, so it deals with modulation automatically, since neighbor islands are independent and can have different tonalities. In addition to this, the rules proposed deals with complex harmonic structures, as modal borrowing, secondary dominant and composite functions. Some of then are not covered by any of the cited works.

4.1. Chord Pattern Matching

This step consists of searching for chord patterns, i.e., recurrent and unambiguous chord sequences. We consider simple patterns, sequences of chords with no complex harmonic structure. Unambiguous patterns are chord sequences that cannot have two possible harmonic analysis. Examples of such patterns are II-V, IV-V-I, etc.

We use a rule base coupled with a first order inference engine, called JEOPS [3] that works on forward chaining. Most of the rules in this base were adapted from Ramalho [10]. The others were created based on the result of a manual analysis of many songs with Jazz and Bossa Nova (a style of Brazilian music) harmony, from composers such as Tom Jobim, Baden Powel, Cartola, Vinícius de Moraes, George Benson, Louis Armstrong, Herbie Hancok, Miles Davis and others.

All the rules apply to consecutive chords, regardless to pitch names, i.e., the rules in this base do not consider the interval between the roots of two chords, but the number of steps between these roots. Once a rule is fired on a set of chords, a new pattern is created with the chords functions. Than, the pattern is inserted in the patterns list. Figure 3 shows how rules of this base looks like.

```
rule II-V Major {
  declarations
    c1 and c2 are chords from the knowledge base;
  conditions
    c1 preceeds c2;
    c1 is minor and has a minor seventh, in case it has;
    c2 is dominant;
    the interval between the roots of c1 and c2 is
        five steps;
  actions
    create a new pattern c, whose tonality is major
        and whose root is a perfect fourth interval
        above the root of c2;
    insert function "IIm7" on pattern c;
    insert function "V7" on pattern c;
    insert c on the knowledge base;
}
```

Figure 3. Major II–V pattern recognition rule.

There is no priority among rules, i.e., to any set of chords, if more than one rule apply, all will fire, as shown in Figure 4. We treat the ambiguous cases in the third step of analysis.

As one can imagine, after running the inference engine, the pattern list will contain many overlapping patterns, as in the example of Figure 4. However, this kind of redundancy is necessary to avoid information loss when eliminating lesser priority patterns. In the example on Figure 4, if the pattern 4 is eliminated in order to insert Am in another pattern, the functions of the other chords are not lost, because of the patterns 1, 2 and 3 that keep the information available.

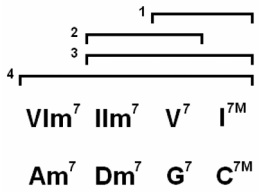

Figure 4. Overlapping patterns in excerpt of a song. Each line represents a pattern found.

The predefined patterns in this rule base follow the Table 1.

4.2. Overlapping Patterns Removal

After applying the first rule base to match the pre-defined patterns, the pattern list may contain overlapping patterns (as shown in Figure 4). The goal of this analysis step is to eliminate all the overlaps, i.e., delete patterns so that the pattern list has no more than one function per chord. To reach this goal, the rules of this base eliminate from the knowledge base the patterns considered to have lower priority or to be redundant.

Table 1. Simple Pattern Matching rules.

Minor Harmonic Patterns	Major Patterns
bII^7 Im	V^7 I^{7M}
IVm^7 Im	IIm^7 V^7
$IIm^{7(b5)}$ bII^7 Im	bII^7 I^{7M}
V^7 Im^{7M}	IV^{7M} I^{7M}
VII° Im^{7M}	IIm^7 V^7 I^{7M}
$IIm^{7(b5)}$ V^7	IIm^7 bII^7 I^{7M}
$IIm^{7(b5)}$ V^7 I^{7M}	VIm^7 Iim^7 V^7 I^{7M}
bVI^{7M} $IIm^{7(b5)}$ V^7 Im^{7M}	IV^{7M} $IIIm^7$ VIm^7 IIm^7 V^7 I^{7M}
IVm^7 $bIII^{7M(\#5)}$ bVI^{7M} $Iim^{7(b5)}$ V^7 Im^{7M}	

There are fourteen rules in this base that were defined from the possible overlaps generated by the first analysis step. In a deeper analysis, we can notice that there are two types of overlaps: total – in effect, a pattern is totally inside another one – or partial with an unique common function – this is, the last function of the first pattern coincides with the first function of the second pattern. Both cases are shown in Figure 5.

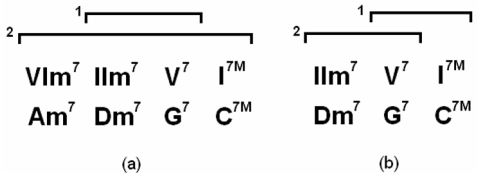

Figure 5. Total overlapping (a) and partial overlapping (b).

We had defined two types of rules: lower priority patterns elimination and redundant patterns elimination. The former type regards to partial overlapping, deleting lower priority pattern from the knowledge base, and the latter deals with total overlapping, preferring the biggest patterns. The rules defined cover more cases than Pachet [8] and Ramalho [10].

We use a reasoning control mechanism to attribute different priorities to the rules of the base [3], i.e., once a rule has fired to a set of patterns, no other rule will fire over the same set of patterns. In Table 2, we define the rules of this base and the eliminated pattern in each case. Rules like "Overlapping A – B" means a partial overlap in which the last chord of the first pattern has a function A and the first chord of the second pattern has a function B, regardless to the functions of the other chords of both patterns.

Table 2. Overlapping removal rules.

Rule	Deleted Pattern	Rule	Deleted Pattern
Overlapping Im – VIm^7	A	Overlapping V^7 – bII^7	A
Overlapping Im – IIm^7	A	Overlapping V^7 – V^7	A
Overlapping Im – IVm^7	A	Overlapping I – V^7	A
Overlapping Im –Vm^7	A	Overlapping I – bII^7	A
Overlapping Vm^7 – VIm^7	B	Overlapping I – IV^{7M}	B
Overlapping Vm^7 – IIm^7	B	Overlapping I – bVI^{7M}	B
Overlapping Vm^7 – IVm^7	B	Subsume	Smaller pattern

4.3. Gaps Classification

Finally, the third rule base has the goal of trying to classify the unclassified chords, i.e., to identify the function of the chords that are out of any pattern. The rule base contains twenty-eight rules, divided into eight groups, and organized in a priority order. Once a rule has been fired to a chord, no other rule can fire to the same chord, since the chord is not a gap anymore.

This step is responsible for recognizing the most complex harmonic structures, as modal borrowing, secondary dominants and composed functions (local modulations as A^7 function in A^7 Dm^7 G^7 C, classified as V^7/IIm, while the whole sequence is classified as V^7/IIm IIm^7 V^7 I).

First, we try to classify the chords according to the harmonic fields of previous and following patterns. If the chord belongs to one of these harmonic fields, it is inserted in the pattern with its respective function in that harmonic field, as shown in Figure 6.

Figure 6. Gap is classified according to its neighbor pattern's harmonic field.

If none of the previous rules fires to the gap, we try to classify it using a composite function by considering the previous and following chords as roots of a harmonic field and trying to find a function that belongs to one of these harmonic fields and can be applied to the gap simultaneously. If the rule fires, the gap is inserted in the pattern to which its neighbor chord belongs, with the composite function associated, as shown in Figure 7. We can see that A^7 belongs to D

minor harmonic field, with the function V^7 (a secondary dominant).

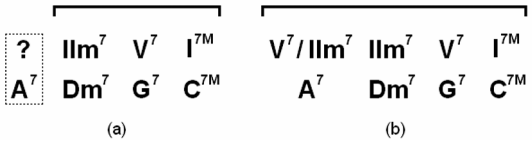

Figure 7. Gap is classified according to its neighbor chord's harmonic field.

Following the priority order of the rules, the next attempt is to classify the gap as a modal borrowing of its neighbor patterns. Figure 8 shows an example of modal borrowing classification. We note that $Dm^{7(b5)}$ belongs to C minor harmonic field with the function $IIm^{7(b5)}$, however the tonality of its neighbor pattern is C major.

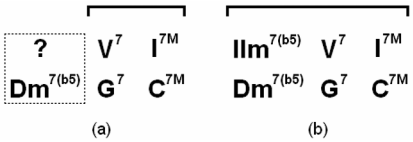

Figure 8. Gap classified as a modal borrowing of its neighbor pattern.

A specific rule was created to classify substitute dominants, i.e., gaps which function is bII^7 relatively to its neighbor patterns, since this function is not identified in any other rule. Figure 9 shows an example of bII^7 classification.

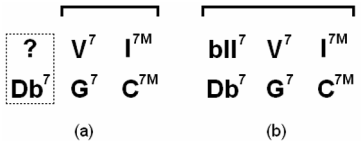

Figure 9. Gap classified as a substitute dominant of its neighbor pattern.

Analogously, when it is not possible to classify the gap as substitute dominant of its neighbor patterns, we try to classify it as a substitute dominant of its neighbor chords, as shown in Figure 10.

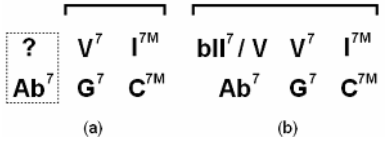

Figure 10. Gap classified as a substitute dominant of its neighbor chord.

Pachet [8] and Ramalho [10] do not consider some of these cases, such as modal borrowing and composite functions.

4.4. Chords Names Correction

There is a big amount of song lyrics mixed with chord notation available on the web. The approach was designed to make use of these inputs. However, web songs often have enharmonic spelling mistakes, as shown in Figure 11.

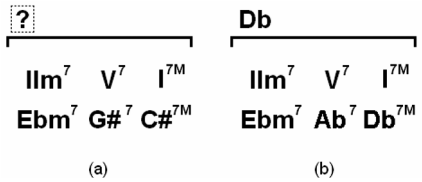

Figure 11. Excerpt of a song before and after chord names correction.

On the example shown in Figure 11 (a), there are two possible tonalities for the excerpt. On the right, (b) there is a possible result for the chord names correction, where the tonality is easily identified.

It is a good feature for the analysis level not to distinguish enharmonic notes, such as $G\#^7$ and Ab^7, since this way the functions may be identified even if the chord grid has mistakes. Because of these enharmonic spelling mistakes, the first level rules are more robust, based only on steps quantity, regardless of pitch names.

Since the system knows what the correct chord names are, based in the analysis made, it corrects the pitch name of the misspelled chords, i.e., in addition to the analysis, even if the input is incorrect, the framework corrects the chord names at the end of the process. All other works cited do not regard to this question.

The rules of this base works as follows: first, we try to use tonalities that had been used before within the same song. Then, if the first attempt is not possible, we use a tonality transition table in order to ease the execution of the music. Finally, each pattern tonality is corrected locally, based in its last chord.

The use of the tonalities transition table has the goal of making it easier to read and execute the music, avoiding the use of complex tonalities or rough tonality changes. The table specifies what tonality, between all enharmonic options, the following pattern may use, according to the current pattern's tonality. The transition table is based on the fifths cycle and avoids double sharpened and double flattened tonalities. For instance, if there is an excerpt of the song which tonality is Gb and the next excerpt has tonality D#, the

last one will be changed to Eb, since the distance from Gb to Eb in the cycle of fifths is smaller than from Gb to D#.

5. Implementation

Our system was implemented in Java, an object-oriented programming language, for some reasons, including the simple integration with useful existing components such as JEOPS [3], a first-order logic inference engine, and Ritornello, in Java [12], a musical framework based on the MusES, in Smalltalk [13]. The designed architecture was built of five basic modules, namely: a parser, a file manager, an analysis engine, a statistical calculator and a GUI. The analysis rules are independent of the implementation, providing means to easily maintain, refine and enrich them.

The Parser is responsible for acquiring musical information from a text passed as input. This was designed to take advantage of the amount of songs available on the web, where it is represented, in general, by lyrics mixed with chord notation. Roughly, the Parser algorithm scans the text ignoring the lyrics and identifying chords that comply with the Brazilian pattern.

We have a File Manager to save and retrieve analyzed songs, including an extension to convert to XML format, aiming for integration with future applications. The Analyzer module is built over the classes generated by the inference engine, which implement the defined analysis rules. The framework encapsulates these classes and integrates them with the other modules. The fourth module is the Evaluator, which provides means to evaluate the quality of the analysis made by the system, including comparison with human-made analysis.

The Graphical User Interface (GUI) is completely independent of the internal implementation of the framework, and uses its services. The services currently provided include opening and saving music files (.mus), generating xml files, song analysis (with an option to do it step-by-step) and analysis edition, for the user to correct eventual mistakes on the automated analysis.

6. Results

Our system was built aiming to analyze music whose harmony is functional. The tests were realized with Jazz and Bossa Nova songs, because of their rich harmony.

Although we could apply the automatic analysis to a larger amount of songs, we were very strict and careful when measuring the quality of the analysis made, so we used a set of twelve harmonic analysis hand-made by senior musicians for comparison with the automatic analysis.

The twelve songs of the test set comprise a total of 810 chords, and our statistical calculator showed that the specialist left 3,52% of the chords unclassified, the automatic analysis left 1,76%. However, considering that the specialist analysis is the ideal analysis, 9,16% of the chords were classified by the system with alternative functions. Considering this context, the framework has reached good results with this set of songs.

During the tests, we could perceive that alternative analysis may imply in chained errors, meaning that, if a given chord was analyzed with an alternative function, the chords related to it may be affected, making bigger the undesired effect and harming the final results.

7. Conclusions and Future Work

We have taken advantage of previous works, mainly Pachet [8] and Ramalho [10], and we have made some significant extensions. More cases are now covered by the pattern matching level, for instance composite functions and modal borrowing. We also added a step for correcting the chord names after the analysis, which did not exist in previous works. Moreover, the implementation comprises completely independent modules, what makes it easy to change the rule basis or use each module separately.

The knowledge acquisition task performed with the experts, necessary to provide the rules of the four rule bases used by the harmonic analyzer, was one of the hardest tasks in this study. There is no formal definition of such rules available on the literature, and the nature of the analysis is often subjective.

Another improvement may be the identification of suspect chords, meaning gaps or chords classified with complex alternative functions regarding to the context. These suspect chords occur due to existing mistakes on chords in music from the web, mainly with chords build over the same notes. The successful identification of this suspect chords and substitution for more appropriate equivalent ones would help the analysis. For instance, consider the sequence Dm/B E^7 Am^{7M}. It has a mistake in its first chord: if we consider $Bm^{7(b5)}$ in place of Dm/B, the analysis would be more coherent, since the functions would be $IIm^{7(b5)}$ V^7 I^{7M}.

10. References

[1] D. L. Baggi, “NeurSwing: A Connectionist Workbench for the Investigation of Swing in Afro-American Jazz”, *Signals, Systems and Computers*, USA, pp. 336-341, 1989.

[2] J. J. Bharucha, “Pitch, harmony, and neural nets: A psychological perspective”, *Music and Connectionism*, The MIT Press, USA, 1991.

[3] C. S. Figueira Filho, G. L. Ramalho, “JEOPS - The Java Embedded Object Production System”, *Advances in Artificial Intelligence. Lecture Notes on Artificial Intelligence Series*, Springer-Verlag, London, UK, vol. 1952, pp. 52-61, 2000.

[4] F. Giomi, and M. Ligabue, “Computational Generation and Study of Jazz Music”, *Interface*, vol. 20, pp. 47-63, 1991.

[5] C. L. Krumhansl, *Cognitive foundations of musical pitch*, Oxford University Press, New York, USA, 1990.

[6] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*, The MIT Press, USA, 1983.

[7] E. Narmour, *The Analysis and Cognition of Basic Musical Structures: The Implication-Realization Model*, Chicago University Press, USA, 1990.

[8] F. Pachet, “A Meta-Level Architecture Applied to the Analysis of Jazz Chord Sequences”, *Proceedings of the 1989 International Computer Music Conference*, Montreal, Canada, 1991.

[9] F. Pachet, “Computer Analysis of Jazz Chord Sequences: Is Solar a Blues?”, *Readings in Music and Artificial Intelligence*, Harwood Academic Publishers, Paris, France, 2000.

[10] G. L. Ramalho, *Construction D’un Agent Rationnel Jouant Du Jazz*, PhD Thesis, Université Paris VI, France, 1997.

[11] A. Chediak, *Songbook Bossa Nova*, Vol. 1-5, Lumiar Editora, Brazil, 1990 (in Portuguese, can be found in web stores such as Amazon).

[12] S. P. Serapião, *Ritornello: Um Framework para Representação do Conhecimento Musical*, Master Thesis, Universidade Federal de Pernambuco, Brazil, 2004 (in Portuguese).

[13] F. Pachet, G. Ramalho and J. Carrive, “Representing Temporal Musical Objects and Reasoning in the MusES System”, *Journal of New Music Research*, Swets & Zeitlinger, Amsterdam, 25(3), pp. 253-273, 1996.

[14] J. Sabater, J. L. Arcos and R. L. Mántaras, “Using Rules to support Case-Based Reasoning for harmonizing melodies”, *Proceedings of the AAAI Spring Symposium Series Multimodal Reasoning*, AAAI Press, California, USA, pp.147-151, 1998.

[15] M. J. Steedman, “A Generative Grammar for Jazz Chord Sequences”, *Music Perception*, Vol. 2, No 1, Scotland, UK, pp. 52-77, 1984.

[16] D. Temperley, D. Sleator, “An Algorith for Hamonic Analysis”, *Music Perception*, Vol. 15, pp. 31-68, 1997.

[17] D. Temperley, *The Cognition of Basic Musical Structures*, The MIT Press, USA, 2001.

[18] J. W. Ulrich, “The Analysis and Synthesis of Jazz by Computer”, *Proceedings of The Fifth International Joint. Conference on Artificial Intelligence*, Morgan Kaufman, California, USA, pp. 865-872, 1977.