A Logical Approach for Melodic Variations

Flavio Omar Everardo Pérez
Universidad de las Américas Puebla
Cholula, Puebla
flavio.everardopz@udlap.mx

Abstract. The use of answer set programming has been applied in a variety of areas and has attracted more and more researchers from different disciplines, one of them is music. Music is an area that has tested the use of answer set solvers for melodic, rhythmic and harmonic compositions with different voices, or even to propose electronic/trance music compositions with more than three instruments. Likewise, the music is governed by rules for their composition, which is why in this paper it is introduced the use of "Aim", a system that uses an answer set approach to modify and propose melodic variations according to known musical structures.

Keywords: Answer Set Programming, AIM, melodic variations, music composition, logic

1 Introduction

From complex knowledge representations, reasoning, learning techniques and son others, the AI has now a great amount of resources to manipulate and process musical material. The music and the AI have been working together for a long time from automatic composition systems, improvisation algorithms, systems dedicated for expressive performances and so many other areas in which researchers have found a place of opportunities [17]. One of these areas is the musical variations or even more, melodic variations. In this paper I propose a system called AIM (Alterations In Music) which uses rule-based techniques to provide alterations in a single melodic line. Rule-based systems have been used in computing for a long time and are the basis for knowledge-based or 'expert' systems used in AI [1]. The process of musical production consists of two major steps according to [12] which are: composition and arrangement. This paper will focus on a type of arrangement varying melodic fragments.

2 Background inside the alterations in musical material

The variation in musical material is an extended area due to the large amount of variables that can be changed while processing musical data. Some related works proposed variations of timbre, phrases, volume, samples, instruments and so many others including audio extraction to determine variations about similar compositions. A proper manner of classification of these works about musical variations can be represented into two: the ones who propose musical variations and the works that uses such variations for different purposes but does not contribute to the modification of a piece. The latter show us mainly two works that takes some musical variations to: identify different versions of the same musical compositions [10] and identify similar segments in music even the presence of large variations in certain parameters like dynamics, timbre, execution of note groups, modulation, articulation and tempo progression [14]. Both works uses a given composition (or its variation) for the search of their respective goals but none of them provides a musical variation (or musical execution or performance) of a given piece.

The main goal of this paper is to provide melodic alterations of a given piece and some approaches are shown in here. Primarily musical or even more, melodic variations creates certain questions about the musical elements or notes that may vary and it is important to consider which notes should stay fixed [9]. According to this work, Cheng provides a alternative to produce melodic variations according to cultural influences. First it is necessary to extract the melodic progression, then depending on the chosen cultural style, maps the uncover progressions. The cross-cultural mappings carries themselves the approaches or “rules” that figures out which of the melodic properties in the underlying progressions will preserve and which will be transformed. The last step consist on adjust the melodic surface of the variation to strengthen its resemblance to the theme. With this approach the essence of the piece can remain still and only creates variations of certain aspects.

Another approach for musical variations is shown in [5]; musical variations from a chaotic mapping. This consists in the creation of musical variations based on an original piece by chaotic trajectories. Starting on a pitch sequence, the trajectories are launched between neighbors and the reference and the mapping is designed to track the reference and tempering the extent of the separation between the trajectories. Tracking means that pitches in the variation appear exactly where they did in the source and the mapping links the variation with the original piece by ensuring only the pitches found in the source piece that comprise the variation no matter the trajectories.

Changing the timbre and phrases in existing musical performance as the user wants [16] is clearly one main goal in music alterations. This work proposes a manipulation method that changes the timbre and phrase of an existing instrumental performance by adding new score information to the piece. By changing the phrase it means that the score of a single melodic part will have certain variations. To do so, the system extracts and estimates the tones, and then it calculates the musical deviation and finally renders the final result.

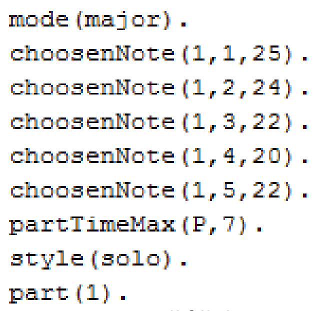
2.1 Considerations about musical variations

The main aspect to think about clearly is to determine the degree of the variations for the listeners to identify that the theme still relies inside the musical variation [9]. This means that how much a piece can vary from the original and still affirm that the essential parts (or the essence of the song) remain. Also it is possible to create a new variation from a given piece and then generate a new variation of the resultant piece. It can be an iterative process [5]. Can a piece still recognizable if it has change from one hearing to another?
All the aspects mentioned above can be summaries in the next question: Which aspects of given piece can be varied and still retaining its identity and which can (or must) stay fixed?

The variation degree that causes that a musical variation is affected in a significant manner or not, can be seen as a computational task rather than a musical one. I consider that it is important to determine a value (possibly given by the user) that can be the parameter to calculate or estimate the number of notes and which notes must change in a melodic line. An interesting approach can be if the variation of a piece cannot differ from the original from more than a 50 percent. With this strategy it can be guaranteed that the resultant piece must have at least some tones that correspond to the original one. Also it is important to limit this value because if not, an interesting question arises as: What would be the difference between composing a new song and varying a piece in a very high significant degree? If we keep only the fundamental note and the rest of the notes given in a small fragment can change, the result can be like composing a new musical piece in a fixed tonality.

3 Rule-based compositions and modifications

Anton [2] is a melodic, harmonic and rhythmic composition system which uses Answer Set Programming (ASP) to generate short musical compositions. Anton uses their own composition engine to propose new fragments or completing a piece. This is made by receiving an input file (in a logic programming format) and the system will fill the spaces until the composition is complete. If the given piece is complete, Anton will provide a single solution (answer set). Anton completes a piece by checking the number of steps (or time) given for the composition and selects the current note according to the previous note or state like shown in Figure1.



```
mode(major).
choosenNote(1,1,25).
choosenNote(1,2,24).
choosenNote(1,3,22).
choosenNote(1,4,20).
choosenNote(1,5,22).
partTimeMax(P,7).
style(solo).
part(1).
```

Figure 1 completing a piece. Anton will fill the spaces for the time 6 and 7

According to the Figure1 Anton will check every given note according to the scale or mode. The time is sequential starting in one and finishing at a given integer value, when a step or time value is missing, Anton will check the previous note and will pick rather a step or a leap. This step or leap value will be added to the note value in the previous state to calculate the current note where the step is missing. With this approach many compositions can be created by a single input according to the possible and valid values that can fill the space. Anton can provide several solutions when the number of missing spaces increases.

Anton can work as a music composition system or as a system that can provide an alteration if a space is required to be filled. At a first instance given Anton a fragment to be completed; it is possible to believe that the system can create melodic variations. One of the main differences is that AIM’s approach to change a note is based on the current state and no matter what note was played in the previous state. With this it can be assure that a melody can be change independently the other notes but respecting the tonality, the scale and the progression of the fragment.

4 AIM – Alterations In Music: System Description for Melodic Lines

AIM is an algorithm dedicated to modify a single melodic line with the use of some compositional rules. This algorithm uses some characteristics provided by Anton and some new features for what the music and melodic alterations are

about. These melodic variations pass through some rules that guaranties valid alterations depending on the tonality of the given piece.
The music composition is governed by rules [3] and according to that, creating alterations requires some rules that can control an outcome by determining the notes that will be changed among executions.

4.1 System Architecture

The AIM software can be divided into two sections: the logical and the sound sections. The logical section describes all the requirements that must be completely achieved to provide an answer set. The musical section is in charge to parse the results to musical notation with the use of Csound [4]. Each answer set corresponds to a new melodic variation from a given input file. To generate a new variation just ask for a new answer set.
The logical section is managed by a driver file called *alterDriver* which receives four parameters for the melodic variations:

- Fundamental – any of the seven notes that corresponds to the C major scale (C, D, E, F, G, A, B).
- Mode – major mode is available for this first version.
- BPM – any value between 120 and 140. This parameter is used like in Armin [6] to calculate the duration of the notes; in other words, this parameter is used to calculate the speed of the song.
- Piece – the input file that corresponds to the melodic line in ASP code. The alterations will be applied to this melodic line.

The *alterDriver* also is the file in charge to call to all the processes for a single run of AIM. This means that this file calls and generates the answer set that corresponds to a melodic variation, also calls the parsing file which maps the model into Csound notation, render the Csound file and play the resulting song.

The input file like is shown in Figure 2 consist on a maximum step value, the number of the notes that will change, a set of notes and their respective durations. The maximum steps value is set to provide AIM the ability to estimate the variations from a given piece. Also it must correspond to be the last note for a given input. The number of notes that can be changed is set as an integer value from one to *n* depending also to the given input. For this example in Figure 2 the maximum valid value is two; this value will be explained more carefully in the further reading. The chosen notes are given as it follows: chosenNote(P,N,CT). It means that the note *N* will be played in order from one to five (*CT* works as a counter). The duration of each note corresponds to the next statement: duration(ST,DR,M,CT) and means that the first note given with the value of *CT* starts in the first measure and last 16 “times”. When the first measure ends, the second one starts again with the start value (*ST*) in one and the duration of the note (*DR*). The note durations were taken from the same scale Armin uses for the trance music compositions [6]; this scale consist in 32 pulses per measure and means that it is possible to fill a measure with any combination of half notes (16), quarter (8), eight (4), sixteenth (2) and thirty-second notes (1). An example is shown in Figure 3.

```
partTimeMax(P,5).

%%Change notes
numberOfNotesToChange(1).
toChangeNumber(1..CN) :- numberOfNotesToChange(CN).

%% choosenNote(part, note, counter).
choosenNote(1,25,1).
choosenNote(1,24,2).
choosenNote(1,22,3).
choosenNote(1,20,4).
choosenNote(1,22,5).

%% duration(start, duration, measure, counter).
duration(1,16,1,1).
duration(16,8,1,2).
duration(24,8,1,3).
duration(1,16,2,4).
duration(16,16,2,5).
```

Figure 2 Fragment of code of the input file

This first version of AIM is dedicated to vary short melodic fragments; this is why this version is limited to only two measures. Also it was decided to fix the number of measures to have more control in the input and in the output files. To have control in the input file is to verify that the input corresponds exactly to two measures, respecting their structure and the measure capacity for allocating notes. Having control in the output file corresponds by checking the integrity of the measures (capacity of the notes) and for testing that the variation was performed in a right way.

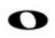
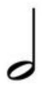
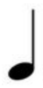
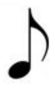
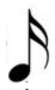
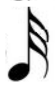
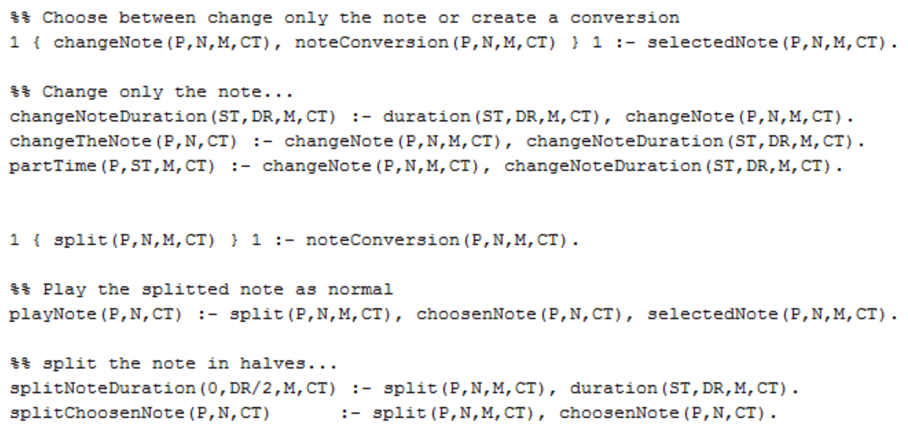
Name	Note Symbol	Musical Value	Duration in Pulses/Time
Whole Note		1	32
Half Note		1/2	16
Quarter Note		1/4	8
Eight Note		1/8	4
Sixteenth Note		1/16	2
Thirty-Second Note		1/32	1

Figure 3 Scale used to determine the duration of each note

Like in Armin, this system uses some codes from Anton's version 2.0.0. These files are: notes, solo, modes and progression. The first three files are used like in Armin or in Anton but with a few modifications. The progression was adapted to work as the engine for the melodic variations. This file is in charge to determine the new note(s) that will be played in the alteration. The way how this progression file works is shown in the next section.

5 The melodic variations engine

Basically a melodic variation consists in deciding which notes will change; in other words it is equivalent to state that it is possible to visit all the notes (starting from the first note) and decide whether the note will apply for a change or not until there are no more notes remaining. AIM’s approach for determining the process to select which notes will change depends directly to the value given as the number of notes to change. If the number equals to one, it means that in all the short melody fragment only one note will be affected. The number of candidate notes to change can vary but there are certain rules that control the behavior of the outcome.



```
%% Choose between change only the note or create a conversion
1 { changeNote(P,N,M,CT), noteConversion(P,N,M,CT) } 1 :- selectedNote(P,N,M,CT).

%% Change only the note...
changeNoteDuration(ST,DR,M,CT) :- duration(ST,DR,M,CT), changeNote(P,N,M,CT).
changeTheNote(P,N,CT) :- changeNote(P,N,M,CT), changeNoteDuration(ST,DR,M,CT).
partTime(P,ST,M,CT) :- changeNote(P,N,M,CT), changeNoteDuration(ST,DR,M,CT).


1 { split(P,N,M,CT) } 1 :- noteConversion(P,N,M,CT).

%% Play the splitted note as normal
playNote(P,N,CT) :- split(P,N,M,CT), choosenNote(P,N,CT), selectedNote(P,N,M,CT).

%% split the note in halves...
splitNoteDuration(0,DR/2,M,CT) :- split(P,N,M,CT), duration(ST,DR,M,CT).
splitChoosenNote(P,N,CT)       :- split(P,N,M,CT), choosenNote(P,N,CT).
```

Figure 4 Fragment of code from the progression file

An important factor to consider about is resumed in the next question: Which notes can be considered to keep the essence of a piece? This work approaches some rules that determine that some notes will remain still. The first, the second and the last note will not change. The reason is because the first note is the fundamental and it can be said that the melody was composed by this starting note. The second note will also remain as it is, to keep the first progression of the fundamental note; no matter if a step or a leap (positive or negative) was chosen, the progression of the first two notes must stay fixed. The last note was decided to remain untouched to preserve the ending of the melodic line and trying to keep some essence of the melodic line. These rules works better when the melody is not to extent or long; in other words, if a short melody is given, it is easier to keep the base of the piece. If a long melody is served as input, maybe more notes will change and the main essence of the piece can vary in a significant way.

Another main functionality inside AIM is the ability to perform two tasks in the melodic variation once a note is already selected for a change. These tasks are: change a note and note conversion. The change a note task means as it name says to only change the selected note. The note conversion part consists on splitting a note into two halves; in other words if the selected note was a half note, it can be divided into two quarter notes (Figure 5). This split conversion is made by dividing the note into its half equivalent; for example if the selected note was a half note (16 pulses) and this note was supposed to play a note *N* (given in the source) it will be divided into two equivalent notes (quarter notes). The first quarter note will play exactly the same note *N* mentioned in the source file and the second quarter note will apply for a change like the change note task (Figure 5).

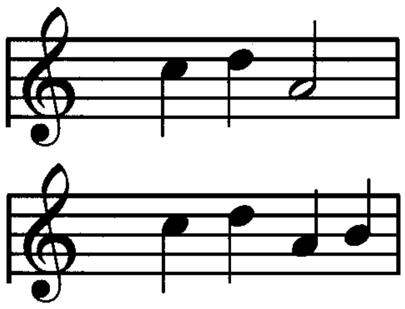


Figure 5 Note conversion. Split a note into halves

The change note process whether comes from a change note task or a second divided note consists by determining rather a step or a leap, positive or negative (positive or negative refers to the chance to play a note lower or higher from the original). It is important to remark that the process of selecting the change note or the note conversion task will only apply for one note at a time; that means that both tasks cannot be applied for the same note at the same time. The code also keeps control from the input file like it was mention above. Errors can occur rather than constraints and with these rules it is easier to control the input coming from the user.

6 Why ASP?

ASP has become a popular approach to declarative problem solving in the field of Knowledge Representation and Reasoning (KRR). It supports a simple modeling language with high-performance solving processes. The main idea behind ASP is to represent a given computational problem by a declaratively logic program. The solutions to such problem are what we known as answer sets [13].

The solver and grounder used for this system is Clasp [7] and Gringo [8] respectively, taken from the Potassco Project Site[1]. Both implementations are currently leading the ASP area.

7 Results

AIM can propose melodic alterations using ASP for small fragments of melodic compositions. This approach can be used as a creative process or as an idea generator [5] where the score can change from one hearing to another. At this level AIM can propose variations for a single melodic line in three ways: changing only one note, convert and split a note or both. Like is shown in Figure 6, only one note was change from the original source and in Figure 7 a note was taken to propose a split conversion. In both figures, there are two melodic lines, the first one is the source and the second one (below the original) is the one that

[1] Potassco official Website: http://potassco.sourceforge.net/

corresponds to the melodic alteration. Both examples were generated using the G major scale[2].

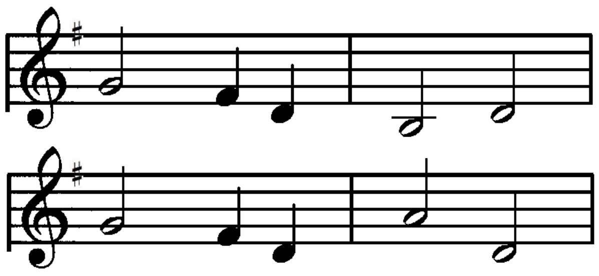


Figure 6 Changing a note in the second measure

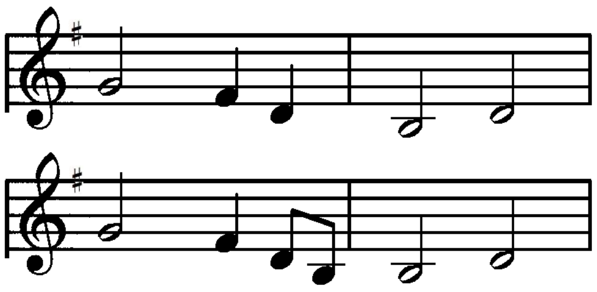


Figure 7 Conversion of one note at the first measure

8 Future work

One of the main functionalities that can help to the users to use AIM is to design and develop another way for introducing the musical data and not from logical code. One solution can be with a user interface with musical sheets and its musical notations, and once the user has completed to fill all the measures, AIM can execute and provide a new melodic variation. This AIM version like it was mentioned before, it provides only the functionality to work with two measures; in other words further versions must allow a big number of musical bars as input for this software. Also an important feature for the melodic variations is the note conversion.

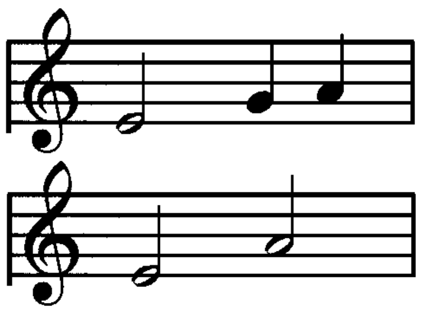


Figure 8 A possible scenario for a note conversion: merging two notes

[2] Examples of executions can be found in http://tranceaiproject.com in the AIM section

AIM provides the ability to split a note but an interesting approach can be the inclusion of the opposite conversion: merge (Figure 8). The merge consists by taking two notes with the same duration and somehow decide which pitch will be played. The duration of the “new” note will be the union of the two selected notes; in other words if two eight notes were chosen for the merge, the new note must last a quarter.

Taking again one of Anton’s functionalities, having more voices (duet, trios, quartets…) to play with, AIM can generate even more variations with harmony between each musical line. Another feature is that given AIM a single melodic line, the system must be able to propose new harmonic lines taking the input as a base for music variations. Finally the inclusion of more modes or musical scales like minor, Lydian, Dorian… for further versions can make the use of AIM a new way to propose full compositions and the inclusion of different volumes assigned to each note or nuances (CSEMPs) [11] for expressive music performances [15].

9 Conclusions

The rule-based approach is an area that has fit in with some musical purposes and music variations can be that area among others. AIM has the flexibility of creating several solutions and proposes new melodic pieces with a single input data. This approach like it was said in Armin [6] it seeks to provide to the user, the chance to discover many unconsidered options as a creative process. An interesting point would be the inclusion of new rules or modifying the existing ones to achieve new variations and also seeking that the essence of the piece will not lose among executions and now old phrases can be replaced with new ones.

References

[1]Andrew R. Brown. 2004. An aesthetic comparison of rule-based and genetic algorithms for generating melodies. *Org. Sound* 9, 2 (August 2004), 191-197. DOI=10.1017/S1355771804000275 http://dx.doi.org/10.1017/S1355771804000275

[2] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2009. ANTON: Composing Logic and Logic Composing. In *Proceedings of the 10th international Conference on Logic Programming and Nonmonotonic Reasoning* (Potsdam, Germany, September 14 - 18, 2009). E. Erdem, F. Lin, and T. Schaub, Eds. Lecture Notes In Artificial Intelligence, vol. 5753. Springer-Verlag, Berlin, Heidelberg, 542-547. DOI= http://dx.doi.org/10.1007/978-3-642-04238-6_55

[3] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2008. Anton: Answer Set Programming in the Service of Music. In *Proceedings of the Twelfth International Workshop on Non-Monotonic Reasoning* (Sydney, Australia, September 13 - 15, 2008). 85-93 http://www.cse.unsw.edu.au/~kr2008/NMR2008/nmr08.pdf

[4] Boulanger, R. (ed.): The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming. MIT Press, Cambridge (2000)

[5] D. S. Dabby. Musical variations from a chaotic mapping. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6, 1996.

[6] Everardo, F. 2011. Armin: Automatic Trance Music Composition using Answer Sets Programming. *XI Encuentro Internacional de Ciencias de la Computación* ENC´11.

Published in *Research in Computer Science Journal* (Marzo 21-25 Toluca Edo. de México) pp.229-237.

[7] Gebser, M., Kaufmann, B., Neumann, A., and Schaub, T. 2007. Conict-Driven Answer Set Solving. In Proceedings of the Twentieth International Joint Conference on Arti_cial Intelligence (IJCAI'07), M. Veloso, Ed. AAAI Press/The MIT Press, 386{392. Available at http://www.ijcai.org/papers07/contents.php.

[8] Gebser, M., Schaub, T., and Thiele, S. 2007. GrinGo: A New Grounder for Answer Set Programming. In proceedings of the Ninth International Conference on Logic Program-ming and Nonmonotonic Reasoning (LPNMR'07), C. Baral, G. Brewka, and J. Schlipf, Eds. Lecture Notes in Arti_cial Intelligence, vol. 4483. Springer-Verlag, 266{271.

[9] Huang, Cheng-Zhi Anna. 2008. Melodic variations : toward cross-cultural transformation. Thesis (S.M.)--Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2008.

[10] J. Serrà. *Identification of versions of the same musical composition by processing audio descriptions*. PhD Thesis. Universitat Pompeu Fabra, Barcelona, Spain. March 2011

[11] Kirke, A. and Miranda, E. R. 2009. A survey of computer systems for expressive music performance. *ACM Comput. Surv.*42, 1 (Dec. 2009), 1-41. DOI= http://doi.acm.org/10.1145/1592451.1592454

[12] Man-Kwan Shan and Shih-Chuan Chiu. 2010. Algorithmic compositions based on discovered musical patterns. *Multimedia Tools Appl.* 46, 1 (January 2010), 1-23. DOI=10.1007/s11042-009-0303-y http://dx.doi.org/10.1007/s11042-009-0303-y

[13] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. 2011. Potassco: The Potsdam Answer Set Solving Collection. *AI Commun.* 24, 2 (April 2011), 107-124.

[14] Meinard Müller and Frank Kurth. 2007. Towards structural analysis of audio recordings in the presence of musical variations. *EURASIP J. Appl. Signal Process.* 2007, 1 (January 2007), 163-163. DOI=10.1155/2007/89686 http://dx.doi.org/10.1155/2007/89686

[15] Miguel Delgado, Waldo Fajardo, and Miguel Molina-Solana. 2011. A state of the art on computational music performance. *Expert Syst. Appl.* 38, 1 (January 2011), 155-160. DOI=10.1016/j.eswa.2010.06.033 http://dx.doi.org/10.1016/j.eswa.2010.06.033

[16] Naoki Yasuraoka, Takehiro Abe, Katsutoshi Itoyama, Toru Takahashi, Tetsuya Ogata, and Hiroshi G. Okuno. 2009. Changing timbre and phrase in existing musical performances as you like: manipulations of single part using harmonic and inharmonic models. In *Proceedings of the 17th ACM international conference on Multimedia* (MM '09). ACM, New York, NY, USA, 203-212. DOI=10.1145/1631272.1631302 http://doi.acm.org/10.1145/1631272.1631302

[17] Ramón Lopez de Mantaras. 2006. Making Music with AI: Some examples. In *Proceeding of the 2006 conference on Rob Milne: A Tribute to a Pioneering AI Scientist, Entrepreneur and Mountaineer*, Alan Bundy and Sean Wilson (Eds.). IOS Press, Amsterdam, The Netherlands, The Netherlands, 90-100.