Armin: Automatic Trance Music Composition using Answer Sets Programming

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

Abstract

The Artificial Intelligence (AI) has taken a leading role in many activities which used to make "by hand", one of them is the musical composition. Such task now has another alternative implementation, through the support of software that can compose different musical genres to the point where a computer can compose pieces with some autonomy. This paper proposes Armin, a system dedicated to the electronic trance music composition. Armin's aim is to provide a trance music composition to serve as a template or a base audio file, ready to add more instruments in a mastering (remastering) production ---Additionally, it seeks to enable greater collaboration between a machine and a human, both seen as composers.

1. Introduction

The Computer science and mathematics provide a large repertoire of algorithms that can be used to generate and process musical material [6] we mean, it is possible to automate certain musical genres based on knowledge obtained through the application rules of its musical components. In this paper we show that it is possible to use answer sets solvers for the composition of short pieces in the trance genre. Armin is named in honor to the great sympathy felt by the author with Armin van Buuren, who stands out as Disk Jockey (DJ), producer inside the trance genre, founder and owner of Armada Music record label. Armin is a system based on Anton [2] with some knowledge of the rules according to trance music, which is able to make a plan and its execution of a etude using Answer Sets Programming (ASP) [10] and at the same time assist the user in the musical production. In addition we show the models inside the trance music and the ASP application in a genre which is characterized by percussive rhythms to set the pace over time. Also we are interest to answer in the future questions like: How much music can be automated? Is there a limit in the music automation? How do you measure the music automation? Is the music automation determined by a number of instruments or if is related to the duration of the piece? During this paper we will talk more deeply about these questions exploring the author's opinions.

2. Background inside the Computer-aided Composition

The Computer-aided composition (CAC) [1] has taken a very important role in composition and musical production, from music pieces which have been created from synthetic instruments (Sound Synthesis) which give lifelike sounds and a wide variety of sound effects to the automatic scores composition. All this avoids the need to go to a studio and record each instrument like before.

The CAC in the same way is known as algorithmic composition. This provides the power to map an idea to software with the options to create and edit the music contents before rendering the final song.

Musicians and scientists have been studied to understand exactly how humans compose music [12] and that have raised questions as "where is the next note coming from?" [3], but the process of selecting the next note is difficult to explain and maybe has some personal taste. Similarly the questions arise once again. If we have already chosen a note, how long must it last? Not only are these questions which we must find the answer, also there are certain considerations to observe in detail because they may pose a problem in the CAC.

2.1 Considerations of Computer-Aided Composition

Music is a finite art where eventually all possible combinations of time and sounds will be explored. A software tool shares the same property, is finite due to the knowledge that counts inside. Although it is possible to get a lot results without repeating any, but we need to think on flexibility and scalability of the system.

One of the difficulties in the CAC is when the "power" is given completely to the tool to produce a complete piece, two cases may occur as mentioned in [12]. Create or imitate. Imitation is given to perceive a result (in this case musical) similar to any other known to us whom we can identify. What would be worth asking is: What rules allow such behavior? Similarly goes the question how much is piece similar to another? Or how many and which notes should vary to conclude that there is a known piece by us? One solution is to find a way get deterministic results by certain rules to provide additional knowledge and maybe some rules that crosses the music boundaries.

With this proposal it is possible to explore a very large amount of results and some fragments that may lead to a "never heard before" by humans or ever even imagined by us or by the pioneers of trance music.

3. Inside the Trance Music

Trance is a genre of the electronic music with a percussion frequency between 130 and 140 Beats per Minute (BPM). The melodies are long and they change or evolve over time. One of the main features of trance is that its time signature is four quarters (4/4) which specifies 4 pulses (beats) per measure. Each beat is identified by a kick (Bass Drum or BD), possibly accompanied at times 2 and 4 with snare sounds (SD) or claps.

Another feature lies in the change of pace that is usually done every two, four or eight bars. However, it is possible that rather given a change of pace, just add or discard drum sounds or other minor instrument. This is what is called progression. Throughout the song we get to hear that the rhythm changes are often strategically placed in certain places and not so random (stochastic) and it is said that the rates are increasing in intensity compared to the previous (also progression).

There are moments in songs where there are no beats, in other words the percussions stops to give importance to synthetic sounds, long chord and a possibly slower paces, this part of the music is known in trance as breakdowns or breaks. These breaks are made to give a break to the beat loops, or to change in an attractive way the rhythm of the song. The beats will later returns to resume the song's tone. A trance song structure may differ from the length of the same, but the essence remains between different versions of the song. Currently the major rankings of electronic music included in its top trance DJs as Armin van Buuren, DJ Tiësto, Markus Schulz, Andy Moor and the same Paul Van Dyk. Today the trance sounds has evolved and have achieved a high degree of sound perception and polyphony.

4. Anton

Anton[1] [2] is a melodic, harmonic and rhythmic composition system which uses ASP to generate short pieces. Anton exists in versions 1.0.0, 1.5.0 and 2.0.0 and their main tasks are:

- Compose
- Diagnosing errors
- Completing a piece

4.1 Composition

Anton offers both melodic and harmonic and also rhythmic composition and is available for the implementation up to four voices (solo, duet, trio and quartet). Anton works in the style of "Rules Palestrina" from Renaissance music in the modes: major, minor, Dorian, Phrygian, Lydian and Mixolydian.

4.2 Diagnosing errors

This section allows detecting errors within a piece. This part also appears in the composition section for generating pieces musically valid or correct.

4.3 Completing a piece

Is possible insert a piece fragment in a logic programming format and select the complete piece option in Anton. If the part is complete Anton returns one solution or model. This model consists of a set of steps number given by the letter T which refers to the time at which the note N should be played in a part P.

The complete system consist of three major phases; building the program, running the solver and interpreting the results. Finally there is an option for playing the audio file generated by Csound at the end of the execution.

5. Answer Sets Programming

ASP is a declarative programming paradigm where a software program is used to describe the requirements which must be completed successfully by solving a specific problem [2], this paradigm is based on stable models (answer set). ASP is a strategy that has knowledge based and constraints in order to prevent undesired executions [10].

Each instrument both melodic and percussion consist in a set of rules which allow us to model some part of a musical piece as shown in Figure 1 and thereby determine their behavior over time. The code inside the Figure 1 is an ASP code and the main feature in here is that plays the mandatory hits at the times 0, 4, 8, and 12 like a common or standard trance rate (counting from 0 to 15). In addition there are 12 possible hits in a sixteenths measure so the other 12 hits should be decided if the hit is made or not, if the hit is done, take amplitude from 4 to 6 and play it. If the stroke is not carried out, it is equivalent to mention that the amplitude of the hit is zero.

```
%%default bassDrum Amplitude = 8
bassDrumAmplitude(8).

%%mandatory BD hits for trance music
mandatoryHit(0;4;8;12).

%% if is mandatory hit... get the amplitude and play it
playDrum(X,A) :- bassDrumAmplitude(A), mandatoryHit(X).

%% pick an amplitude and a non mandatory hit...
1{ playDrum(Y,X) : amplitude(X) }1 :- bassDrumHits(Y).
```

Figure 1 Fragment of Armin Bass Drum Generator

[1] Official Website: http://www.cs.bath.ac.uk/~mjb/anton/

The solver and grounder used for this system is Clasp [7] and Gringo [8] respectively, taken from the Potassco Project Site[2]. Both implementations are currently leading the ASP area.

6. Armin – System Description

Armin[3] is a trance music composition algorithm which takes the composition rules represented declaratively using ASP. These rules have the property to model a desired result within the genre and to explore future extended compositions. Using in part the operation of the Anton for melodic composition, Armin provides the feature to expand into the main percussion and musical sections chaining, like introduction to verse, verse to chorus, chorus to breakdown... in order assemble these sections with some dependency to generate the next state (Markov Chains). One feature that is sought is the ability to perform a "valid" musical composition inside the trance genre with its dependencies of the internal parts of a traditional song and percussion and its respective progression between the different sections. It should be noted that the objective of Armin in its first version is not fully compose a piece of the trance music, but to serve as a "template" both melodic and percussive and other sounds and lately make some post production to this source. Additionally it seeks to explore different outcomes (each answer a set corresponds to a valid piece of music) that have openness within the music style and looking forward to propose variations in its composition completely valid. To generate a new piece of music just ask for a new model or answer set.

6.1 Armin – Architecture and Components

The Armin architecture is divided into two big sections of ASP as shown in Figure 2 which are the Score Generator and the Sounds Generator. It is important to add that there is a strong dependency between these two parties, because the Score's product works as the input of the Sound Generator.

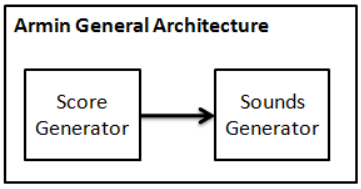

Figure 2 Armin Architecture

The Score Generator consists on a driver and the assembly file as shown in Figure 3. The Armin driver is in charge to manage the tasks during the score execution and is the main file which receives some input parameters, which are:

- BPM – valid values given inside the interval of 130 to 140 including those.
- Fundamental (for melodic instruments) - any of the 12 notes that conforms the chromatic scale.
- Parts / Fragments - this is the number of parts or sections that will be played in the piece. This value is variable depending on the length of the piece that the user wants to generate. The fragments correspond to the parts of a trance song (intro, verses, choruses, breakdowns ...).

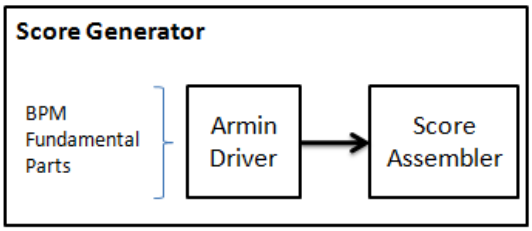

Figure 3 Score Generator

The Score Assembler is an ASP file that generates the order and the appearance frequency of the parts during the resulting song. These parts as mentioned before correspond to the internal structure of a piece. This generation is made by certain dependencies between parts of the piece and its possible behavior over time. Each answer set corresponds to one configuration of a song (this doesn't mean that every time we get the same answer, we will get the same musical result). At this point, we know what that a fragment is going to be played and also we know when is going to happen, but it has not generated any audio until now. The section in charge of this task is Sounds Generator.

Once we have got the score model of the song, this information works as input in the Sounds Generator section which consists of five components as shown in Figure 4, the components are: results interpretation, instruments generator (ASP), Csound parser, render, and finally the final product, an audio file.

The Sounds Generator starts interpreting the obtained results by the model (answer set) in the Results Interpretation section. This file besides understanding the behavior of the song is responsible for sending a call to each instrument that is going to be part of the current structure by calling the Instruments Generator section. This file is responsible for generating the model of each instrument depending on the part or section in the score. Once generated the model of each instrument at a time, is executed the Csound Parser which is responsible for mapping the results to a file in the Csound format. This process takes place until it has reached the total time of the piece or until there are no fragments remaining to be parsed. Once the file is done, is time to run the Csound's

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

[3] Armin official Website: Flavio Everardo: http://flavioeverardo.com in the Automatic Trance Music Composition section.

render and then Csound provides an audio file in WAV format.

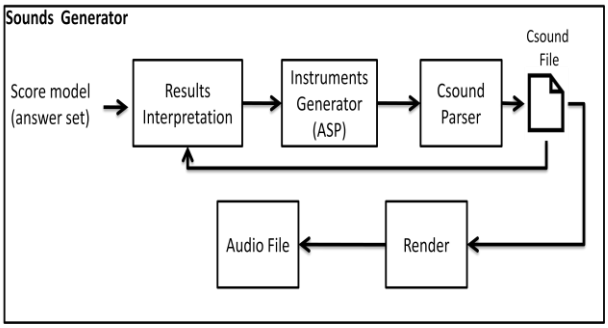

Figure 4 Sounds Generator

The way as an automatic music composition tool creates a musical piece can vary from the goal or essence of the application and in some ways what is evaluated is not the methodology but the final product (song, album, performance ...). This is always true when the objective of the application is public and not a personal domain [Pearce et al. 2002]. If there is a personal reason or purpose behind the application, we can say that there is no exact methodology neither restrictions that limits the production and musical performance.

There are different approaches to the composition of a musical piece which depend on a parameter that is set as a starting or reference point for the production of the piece, such as time or duration, the number of elements within the structure of the song or even the instrument(s) chosen. Armin presents a sequential composition structure taking the particularity that the time in Csound is cumulative, and it takes by default an entry point called introduction (intro). All pieces start off with their respective intro (this doesn't mean that all the pieces will have the same behavior) at time zero and the following states are calculated using the current state. The last fragment is dedicated to the end of the song (outro) which also has its own dependency on the previous state.

For Armin architecture it is used the Potassco software (Clasp and Gringo) as the declarative language, the Perl programming language is used for the results interpretation and parsing and Csound for the audio synthesis.

7. The Automation

We believe that the music can be automated in several ways, which depends on the purpose of the software or the developers. The important question here to determine how much can music be automated is: what are you going to let the user do after delivering a musical piece? In other words what would be the composer's contributions?

The automation cannot be measure in an easy way. Many people will be only cared for the musical results and answering themselves if the system is doing what it is supposed to do. Perhaps in the future many algorithms composition systems will have to adopt a base or standard to be evaluated among them.

8. Results

Armin doesn't provide a finish work; it provides us a base (template) song, ready for adding more instruments in the way the users wants. Armin delivers a musical piece as a starting point to compose trance music. The features inside a base song are the implementation of the bass drum, hi hats, snare drum and the dependency of this instruments among the time for short pieces (30 – 60 seconds). The melodic part is currently in developing for trance rhythms. Building these short pieces it doesn't mean a problem for the system and we are looking to create a more complete template song. In addition Armin provides openness in the drums generators, respecting the principles of the trance music, but adding some new beats features to give us different kind of songs among executions. At this point because of the length of the song it is not easy to identify the elements of a trance song by hearing, this is because the beats nuances are not completely natural. What we mentioned about the Markov Chains it is currently at a conceptual level; however is in current developing by now. We have beats fragments already in audio format which you can hear and download in Armin's site[4].

9. Future work

9.1 Musical work

Inside the musical work there are a lot of things to do starting with the addition of more instruments suitable for trance music. Also, like the Computer Systems for Expressive Music Performance (CSEMP) [9] Armin can be extended to a more real and not machine performance. This can be reached by working with the nuances inside the notes and beats.

An interesting point inside the trance music would be to model different versions of a generated piece like extended ones, which have the particularity of lasting more than six or seven minutes. Besides creating the variations of a given piece would be an interesting challenge for the taste of every user.

Another feature is the rules testing. Study the ways in which a rule is acting over the others or if it's worth to change the rule or model the rule in a different way to get another and a better meaning. In addition the possibility to get more in touch with professional people inside the trance area will be very grateful. Finally add more rhythms to play with and fills on every two or four

[4] Short pieces and examples can be found in http://flavioeverardo.com in the Automatic Trance Music Composition section.

measure, this will also provide the opportunity to make more "unique" pieces among executions.

9.2 Computational work

Among the computational work we can find in a first order, the use of data mining to find more patterns inside the style and adding a plus inside the "reality" of the musical production. The second and also important will be the user's feedback in a machine learning purpose. Having the chance to "vote" or "grade" the songs quality, compositional structure, notes sequence… can be a useful knowledge for future performances.

Another computational work is the chance that the user can modify and edit every step during the execution in a graphical user interface to make more suitable the human interaction.

10. Conclusions

Music is much to be a personal taste area; this is because we perceive music in several ways. Besides, judging is a subjective process [4] and we all do not like the same music. Armin has the flexibility of creating several solutions with the same score results and of course, it can provide more results if the score model change among executions. This provides the users the capability of discover many unconsidered options by them. Adding more rules and instruments can supply us even more unexpected results and more creative pieces. Besides this allows the composers to create the moldings for his own creations [4] and styles.

11. References

[1] Anders, T. Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System. Ph.D thesis, Queen's University, Belfast, Department of Music (2007)

[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] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2008. Automatic Composition of Melodic and Harmonic Music by Answer Set Programming. In *Lecture Notes in Computer Science*, 2008, Volume 5366/2008, 160-174, DOI: 10.1007/978-3-540-89982-2_21 http://www.springerlink.com/content/rj422j5831w0278m/

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

[6] Dahlstedt, P. 2007. Autonomous evolution of complete piano pieces and performances. In Proceedings of the ECAL Workshop on Music and Artificial Life (MusicAL , Lisbon, Portugal, Sep.), 10.

[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] 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

[10] Lifschitz, V. 2008. What is answer set programming?. In *Proceedings of the 23rd National Conference on Artificial intelligence - Volume 3*(Chicago, Illinois, July 13 - 17, 2008). A. Cohn, Ed. Aaai Conference On Artificial Intelligence. AAAI Press, 1594-1597.

[11] Pearce, M. T., Meredith, D. and Wiggins, G. A. (2002). Motivations and methodologies for automation of the compositional process. Musicae Scientiae, 6(2), pp. 119-147.

[12] Senaratna N. I. 2006. Automatic Music Composition with ACMTIES. University of Colombo School if Computing.