September 2016

Classifying and Generalizing Successful Parameter Combinations for Sound Design

Iván PAZ [1], Àngela NEBOT FranciscoMÚGICA and Enrique ROMERO
Soft Computing Research Group, Computer Science Department, BarcelonaTech

Abstract. Operating parametic systems in the context of sound design imposes cognitive and practical challenges. The present contribution applies rule extraction to analyze and to generalize a set of parameter combinations, which have been preselected as successfully producing a specific perceptual category. Then, it is discussed how and under which conditions these generalizations can be used, for example, for the automation of specific tasks.

Keywords. rule extraction sound design parametric systems generalizing data sets

1. Introduction

Computer music is a growing field. Ever since its origins around the middle of the twentieth century, when the first musical notes were generated by a computer programmed by Alan Turing, new systems for making music by means of computer technology constantly appear. This tendency increased during the last years as personal computers began to be affordable for a wider public (see, for example, references [1] and [2] published respectively, in 2009 and 2018).

Within computer music, sound design is responsible for generating and processing the sound until it acquires the desired characteristics. Generally speaking, this "shaping" process is done through systems of many different architectures (including synthesizers and signal processors). Regardless their specific characteristics, they are all controlled by parameters whose different combinations determine its sonic possibilities. However, such parametric spaces are sometimes so large or complex that, for example, when systems (like sintesizers or signal processors) are sold commercially, a bank of presets (preselected human-categorized combinations) is included to facilitate its use [3,4]. In addittion, when synthesizers or signal processors are implemented in software (a very common practice in fields such as algorithmic composition or live coding), for the musician to know a priori how the different parameter combinations will sound, she would require background in acoustics, signal processing and computer programming. For this reason,

[1]Corresponding Author: C/ Jordi Girona 1-3, Omega 005, 08034 Barcelona, Spain; E-mail: ivanpaz@CS.upc.edu

methodologies helping the user to explore the space, finding successful combinations of parameters satisfying a musical idea in mind have been proposed [5,6]. They use genetic algorithms and interactive evolution [7] which uses human audition to guide the search and to label the combinations with their exhibited perceptual categories (as perceibed by the subject). Examples of perceptual categories can be consonance, dissonance, or more abstract conceptions such as chaos or tension. Although these methodologies allow to explore the space finding successful presets, they do not allow neither the generation of interpretable models describing the relationships between parameter values and perceptual categories, nor induction (generalization) to find new points or regions of the space producing the same acoustic impresion (perceptual category). Achieving this last task would be very useful, since the process of interactive evolution requires the subject to listen and evaluate each of the examples, so the number of presets obtained is limited. For this reason, in [8,9], the authors proposed a methodology designed to work with perceptually labeled presets and to extract rule models with different levels of generalization. The base of presets can come from an assisted search as the proposed in [5,6], or they can be those used in any existing piece. The methodology addresses the following objectives:

- To extract interpretable models of the relationships between the parameter values and the assigned categories.
- To extend the bank of presets, when possible, by including unexplored regions of the space but which are expected to produce acoustic outputs within a chosen perceptual category.
- To facilitate specific musical tasks, such as automate the creation of constant variations during a performance, and to suggest new lines of research. For example, until which point the proposed methodology can support compositional approaches as the one described in [10].

The present contribution presents and discusses the aforementioned objectives throught a concrete case of study, using the Rulex algorithm [9] to analyze and to extend (by including new unheard parameter combinations) an existent set of presets in the context of sound design. The rest of the paper is structured as follows: In section 2 the Rulex algorithm is introduced, section 3 applies the algorithm to a real set of presets and analyzes the extracted rules from the perspective of its aplicability to different musical tasks, section 4 presents the results and the discussion and finally, section 5 introduces the conclusions and presents possible further work.

2. Rulex algorithm

Rulex is a rule extraction algorithm (hence its name) designed to extract patterns form labelled data. In the context of this research, the labels describe specific perceptual categories assigned by the user. The extracted models (the identified patterns) generalize, to different levels, the relationships between the categories and the parameter values. For this, the algorithm structures the input data into patterns defined in terms of the Hamming distance. According to the selected distance, the extracted rules may or may not include new unheard combinations which are assumed to be consistent with their associated label, or category.

2.1. Presets, Rules and Patterns

An input datum is named "preset" evoking the way pre-programmed settings on musical instruments (for example a synthesizer patch) are named. A preset is equivalent to an *example*, in the sense of classification tasks [11], in which the variables are the parameter values of a musical system, and the class is the perceptual category assigned to the combination. The Rulex algorithm takes as input a set of "presets" and searches for patterns. When a pattern among presets is found they are grouped and represented as a single data. The output is a set containing the groupings that occurred, as well as the presets that were not grouped. All the output data is represented in "Rule format". Rules are lists containing sets in each of the entries corresponding to the parameters and a class in the last one. Next, preset, rule and pattern are defined.

2.1.1. Preset

A preset is a labeled parameter combination. It is represented in a list in which the first $n-1$ entries are parameter values, and the n^{th} entry contains the perceptual category (or class) associated with the combination. E.g. $[p_1, p_2, ...p_{n-1}, c_j]$.

2.1.2. Rule

Is a labeled list in which the initial $n-1$ entries contain sets whose elements are parameter values. The sets may contain one or more elements. The n^{th} entry contains a perceptual category. E.g. $[\{p_1\}, \{p_2, p_2', p_2''\}, \{p_3, p_3''\}, \{p_{n-1}\}, c_j]$.

2.1.3. Pattern

Let r_1 and r_2 be two rules of the same category. We say that a pattern between them exists if when comparing their $n-1$ entries corresponding to the parameters, the number of different entries is less or equal than a given distance threshold $d \in N$.

For a complete documentation of the algorithm the reader is refered to [8,9]. Next, its operation is introduced intuitively by means of an example.

2.2. Rule Extraction Example

Consider the set of presets presented in Table I. They describe the values of the parameters of a synthesizer and the perceptual category associated to each combination. The synthesizer consists of a couple of square wave generators, each of which has parameters frequency and pulse width. They are represented respectively, by ***freq1*** and ***freq2*** for the frequencies (both measured in Hz) and by ***width1*** and ***width2*** for the pulse widths of the waves (the pulse width ratio ranges from zero to one, 0.5 making a square wave). The two waves are added and the output is controlled by a general normalized amplitude (***amp***). The SuperCollider implementation is:

```
Ndef(\synthesizer1,{
arg freq1, freq2, width1, width2, amp;
var sig;
sig=Pulse.ar(freq1,width1,amp)+Pulse.ar(freq2,width2,amp)});
```

The labels "intro and break" (at the end of each preset in Table I) are the perceptual categories, representing in this case the section of the piece for which the combinations were selected.

Table 1. Presets describing parameter combinations of a synthesizer and the perceptual category associated to each one

Preset	**Parameter values and perceptual category**					
Number	***freq1***	***freq2***	***width1***	***width2***	***amp***	***Category***
1	82.5	55.25	0.5	0.5	0.23	intro
2	660	221	0.3	0.5	0.23	intro
3	330	221	0.3	0.5	0.23	intro
4	165	218	0.3	0.5	0.23	break
5	660	110.5	0.3	0.5	0.23	intro

The extracted rules when searching for patterns with distances d = 1, 2 and 3 are shown in Table II.

Table 2. Extracted rules from the set of presets presented in Table I, when searching for patterns setting the distance d = 1, 2 and 3

Rule	**Parameter values and perceptual category**					
d	***freq1***	***freq2***	***width1***	***width2***	***amp***	***Category***
d **= 1**						
$r_{2,3}$	{660,330}	{221}	{0.3}	{0.5}	{0.23}	intro
$r_{2,5}$	{660}	{221,110.5}	{0.3}	{0.5}	{0.23}	intro
r_1	{82.5}	{55.25}	{0.5}	{0.5}	{0.23}	intro
r_4	{165}	{218}	{0.3}	{0.5}	{0.23}	break
d **= 2**						
r_1	{82.5}	{55.25}	{0.5}	{0.5}	{0.23}	intro
$r_{2,3,5}$	{330, 660}	{221, 110.5}	{0.3}	{0.5}	{0.23}	intro
r_4	{165}	{218}	{0.3}	{0.5}	{0.23}	break
d **= 3**						
$r_{1,2,3,5}$	{330, 660,82.5}	{221, 110.5, 55.25}	{0.3, 0.5}	{0.5}	{0.23}	intro
r_4	{165}	{218}	{0.3}	{0.5}	{0.23}	break

[a]Subindexes indicate the presets composing the rule.

2.2.1. Extracted Rules Setting $d = 1$

The rules extracted by setting $d = 1$ show a "grouping" for each pair of presets of the same category that differ, one from the other, in the value of one parameter. In this case, it is possible to group presets 2 and 3 because they differ only in ***freq1*** which is respresented by $r_{2,3}$. It is possible to group presets 2 and 5 into $r_{2,5}$. The rules r_1 and r_4 that can not be grouped with this criterion with any of the other presets are rules describing only one example.

Using AND and OR notation $r_{2,5}$ is read in the following way: IF ***freq1*** = 660Hz AND ***freq2*** = 221Hz OR 110.5Hz AND ***width1*** = 0.3 AND ***width2*** = 0.5 AND ***amp*** = 0.23 THEN Category = intro.

2.2.2. Extracted Rules Setting $d = 2$

Analogously when choosing $d = 2$, all the possible groupings are produced among the presets that differ in ≤ 2 parameters. Notice that the algorithm operates sequencially, i.e. adding preset by preset to a set containing the rules and then searching for patterns between the added preset and the rules that exist there. In the example first the rule $r_{2,3}$ is created and then grouped with preset 5 (as $r_{2,3}$ and r_5 differ from each other by 2 parameters) into $r_{2,3,5}$.

Note also that, unlike when $d = 1$, the rules created by using $d \geq 2$ can describe cases that are not found in the original presets. For example, $r_{2,3,5}$ has the combination [330, 110.5, 0.3, 0.5, 0.23, intro] which is not included in the set of presets of Table I. The result of this varies between the following two extremes: either it turns out that the new added combination is perceived as belonging to the category of the rule or not. Both cases and in which systems they occur is discussed in the Section II.D.

2.2.3. Extracted Rules Setting $d = 3$

As said, using greater distances the number of "new" included presets increases. For example, the rule $r_{1,2,3,5}$ contains 18 different combinations of which only 4 are part of the original presets.

2.3. Analyzing the Extracted Rules

Interpreting the extracted rules requires musical knowledge. In particular, the function of each parameter. In the example, $r_{2,3}$ shows that two combinations differing only in ***freq1*** were used in the *intro* part. Furthermore, observing the values in ***freq1*** these were 330 and its respective octave 660. In both cases, ***freq2*** was set to 221. If we think this from a musical point of view, it is expected that if the relation between the frequencies 330 and 221 worked, so does the relationship between 330's octave and 221. In this sense, rules $r_{2,3}$ and $r_{2,5}$ clearly show the use of the octave interval. Considering the function that each parameter has in the definition of sound, we can select which of the extracted rules to use in different contexts. To ilustrate this, consider again the rules of Table II. We have mentioned that rules $r_{2,3}$ and $r_{2,5}$ describe variations in the fundamental frequencies of the square wave generators. Whereas if we observe the rule $r_{1,2,3,5}$, we see that it also shows variations in parameter ***width1***, which modifies the pulse width (remember that ***width*** = 0.5 defines a square wave). Varying the pulse width modifies the timbre of the signal but not its fundamental frequency (as does the ***freq*** parameter). Such musical knowledge allows us, when choosing the rules, to know which parts of these produce timbral variations and which harmonic variations. In this way, the algorithm finds patterns, that would go unnoticed or require hours of work to be identified in larger sets of presets, and presents such patterns to the user as a set of rules. Then, with the necessary knowledge the user can interpret the patterns and select the preferred rules.

2.4. Generalization Using Parameter d

As mentioned, when using distances $d \geq 2$, presets that are not found in the database are included in the rules. This is a consequence of the algorithm design. However, not in all the systems the new presets included are perceived as "belonging to the category of the rule" when being heard by the user. To answer a priori in which systems this

generalization will be effective and in which not is impossible since this depends on the perceptual characteristic chosen by the user (and therefore subjective). However, a simple example can help us to see that it is possible to have some intuition, i.e. once a system is implemented and a perceptual feature chosen, the user can intuit what level of generalization will work.

Let us consider the additive synthesis described in section 2.2 and a band limited impulse oscillator consistsing of a fudamental frequency to which a certain number of upper harmonics (all with the same amplitude) are added. The supercollider implementation of this synthesizer2 is as folows:

```
Ndef(\synthesizer2,{
arg freq, numharm;
var sig;
sig = Blip.ar(freq, numharm)})
```

It has tree perceptual regions (described in [12]) that can be described as perceived like rhythmic, rugged and tone. User test using such perceptual categories are described in [8]. A group of presets for each of these categories is shown in Table III. If we analyze

Table 3. Presets describing parameter combinations for synthesizer2 for perceptual categories rhythmic, rugged and tone

Preset	**Parameter values and perceptual category**			
Number	***freq***	***numharm***	***amp***	***Category***
1	11.354432	20	0.6	rhythmic
2	10.203962	20	0.6	rhythmic
3	5.504405	20	0.6	rhythmic
4	1.854298	230	0.6	rhythmic
5	7.653983	230	0.6	rhythmic
6	15.012693	230	0.6	rhythmic
7	4.294679	230	0.6	rhythmic
8	20.425354	260	0.6	rugged
9	24.548191	260	0.6	rugged
10	21.10586	260	0.6	rugged
11	21.10586	67	0.6	rugged
12	21.10586	370	0.6	rugged
13	21.10586	26	0.6	rugged
14	99.598908	7.928433	0.6	tone
15	99.598908	14.141092	0.6	tone
16	55.781054	7.807612	0.6	tone
17	55.781054	1.90927	0.6	tone
Rules using d = 1				
$r_{1,2,3}$	{5.504405, 10.203962, 11.354432}	20	0.6	rhythmic
$r_{4,5,6,7}$	1.854298, 4.294679, 7.653983, 15.012693	230	0.6	rhythmic
$r_{8,9,10}$	{20.425354, 21.10586, 24.548191}	260	0.6	rugged
$r_{11,12,13}$	21.10586	{26, 67, 260, 370}	0.6	rugged
$r_{14,15}$	99.598908	{7.928433, 14.141092}	0.6	tone
$r_{16,17}$	55.781054	{7.807612, 1.90927}	0.6	tone

from a musical point of view the tables I and III, it can be seen that, while in the case of

synthesizer1 the variations in ***freq1*** and ***freq2*** describe octave intervals (defined by the relation $f' = 2f$), in the case of the synthesizer2 the ***numharm*** parameter (that defines the number of upper harmonics added to the fundamental frequency) does not drastically change the perception of the output, which is defined by the value of ***freq***. Therefore it is expected that rules extracted using larger values of distance (d) will be more effective in the case of the synthesizer2 than in the case of synthesizer1. I.e. The perception does not depend on the interrelation of the parameters but rather it is defined by the ***freq***. Why to keep the second parameter during the rule extraction? Because of the timbre variability that it produces. Adding new combinations increases the variability of the timbre in this case. For the rules that add new combination among parameters in the synthesizer1 to be effective, it is necessary that the added combinations among ***freq1*** and ***freq2*** produce effective results, for example, consonance if this were the case.

3. Applying the algorithm to sound design preset sets

This section presents how the Rulex algorithm can be used, given an existing dataset, to analyze it and to explore possibilities of recombining/extending the existing material. The cases in which it is possible to add variability to the existing presets are pointed out, as well as those in which the algorithm suggests new regions to be explored, predicted to be consisentes with some category. Also, it is discussed how the user needs to analyze the possibilities and decide how to use the extracted rules. Keep in mind that the idea of the algorithm is to extend the composer capabilities and not to replace them. In this sense, the algorithm deliberately seeks to detonate synergies by interacting with the user rather than purely returning a non-interpretable or non-modifiable model performing a single task.

3.1. Synthesizers

The synthesizers from which the parameter combinations were selected are the synthesizer1 (same as in section 2.2) and synthesizer3 (shown below). Their SuperCollider implementation is:

```
Ndef(\synthesizer3,{
arg freq3_1, freq3_2, amp3;
var sig;
sig=Mix(Saw.ar([freq3_1,freq3_1+1], amp3)+Saw.ar([freq3_2,freq3_2-1], amp3))});
```

Synthesizer3 is composed of two sawtooth wave generators controlled by ***freq3_1*** and ***freq3_2*** respectively. One more sawtooth with frequency ± 1 Hz is added to each generator. The synthesizer has a general amplitude ***amp3***. Table IV shows a set of presets selected for synthesizer 1 and 3 as successful combinations for the sections denoted by their classes.

Table V shows some rules extracted from the presets of Table IV using different distances.

Table 4. Presets describing parameter combinations of a synthesizers 1 and 3. The perceptual category associated with each combination represents the part of the piece in which the combination is intended to be used

Preset	**Parameter values and perceptual category**								
	freq3_1	***freq3_2***	***amp3***	***freq1***	***width1***	***freq2***	***width2***	***amp***	***Cat***
1	200	159	0.2	161	0.4	160	0.5	0.27	Part A
2	200	150	0.23	150	0.3	160	0.5	0.3	Part B
3	200	150	0.26	150	0.3	160	0.4	0.3	Part B
4	200	150	0.23	150	0.3	161	0.5	0.3	Part B
5	20	150	0.25	101	0.3	102	0.5	0.33	Part C
6	200	150	0.28	150	0.6	160	0.4	0.38	Part C
7	100	100	0.25	100	0.6	102	0.4	0.33	Part C
8	20	150	0.25	101	0.3	102	0.5	0.33	Part D
9	200	150	0.28	150	0.6	160	0.4	0.38	Part D
10	100	100	0.25	100	0.6	102	0.4	0.33	Part D
11	20	150	0.25	101	0.3	102	0.5	0.33	Part E
12	200	150	0.28	150	0.6	160	0.4	0.33	Part E
13	100	100	0.25	100	0.6	102	0.4	0.33	Part E

Table 5. Examples of the rules extracted, using different distance values, from the presets of the Table IV. Note that not all the rules extracted by the model are shown, only those that best illustrate the behavior we want to highlight have been chosen.

Preset	**Parameter values and perceptual category**								
	freq3_1	***freq3_2***	***amp3***	***freq1***	***width1***	***freq2***	***width2***	***amp***	***Cat***
d = 3									
$r_{2,3,4}$	{200}	{150}	{0.23, 0.26}	{150}	{0.3}	{160, 161}	{0.5, 0.4}	{0.3}	Part B
d = 5									
$r_{12,13}$	{200, 100}	{100, 150}	{0.25, 0.28}	{100, 150}	{0.6}	{160, 102}	{0.4}	{0.33}	Part E

3.2. Analyzing and Selecting Rules for Specific Musical Tasks

The extracted rules can be used to solve different musical tasks. For example, to produce slight automated variations, while live performing. This can be done by selecting a rule (or a set of rules), producing parameter combinations by taking an element from each of its sets, and setting them in the synthesizer. Next, the rules shown in Table V are analyzed, so that the reader can visualize how the selection process is carried out. Of course, listening to the rules and selecting those that produce the expected results is the way to do it, however the type of reasoning we show below suggests were to start when there are many rules. To automate this process, signal descriptors could be used to indicate proximity between the produced results. However, for small sets of presets dedicated users prefer to inspect the material. Analyzing the rule $r_{2,3,4}$ (obtained by using d = 3), it can be seen that the parameters in which the sets have more than one element are: ***amp3, freq2*** and ***width2***. Analyzing the function of these parameters in the context in which they are used, it can be said that: The variability that the parameter ***amp3*** will produce, affects only be the volume of synthesizer3, while the variability that parameter ***width2*** will produce affects the timbre of synthesizer1. Then, it is expected that the parameter ***freq2*** will

produce the greatest variations (from a harmonic point of view) when interacting with the other frequencies. However, we can see that all the combinations that can be formed among the frequencies already appear in the original set of preset. So from this rule, if accepted, we would expect it to produce slight variations of timbre and amplitude.

When analyzing the rule $r_{12,13}$ we see that the parameters with sets having more than one element are ***freq3_1, freq2_2, amp3, freq1*** and ***freq2***. In this case, unlike the previous one, although some combinations of parameters (given the architecture of the synthesizers) do not generate strong variations, exchanging the values of ***freq3_1*** and ***freq3_2***, will produce combinations, for example those that can be formed with freq2, that do not appear in the original set of presets. Among these combinations is [100, 100, 0.28, 100, 0.6, 160, 0.4, 0.33, Part E]. Would be the combination of ***freq3_2*** = 100 with ***freq2*** = 160 effective? This will depend on how the perceptual category sought relates to the relationship between these frequencies, which is decided by listening or signal analysis.

4. Discussion

The extracted rules together with the knowledge of the functions (in this case in the context of the sound design) of the parameters of the synthesizers, offer an interpretable model that also allows the user to generalize the material already classified. The different levels of generalization of the rules condition the contexts in which they can be used. For example, in the search for new musical material large distances can be used. While in the context of a live performance it is likely that smaller distances are preferred to only automate the material generation. The described approach has been tested in different contexts. User tests, using a previous version of the algorithm that only extracts rules with d = 1, were performed to verify if the extracted rules offered an interpretable model of the relationships between parameters and perceptual categories [8]. The test yielded successful results. These tests were performed by thinking of the algorithm as a system intended to be used by others appart from its programmer as suggested by [13]. The same version of the algorithm used in a compositional context can be found in [14]. Again accoring with [13] when a composer makes an algorithm to compose music, this can only be evaluated in the way that music composition is traditionally evaluated. That is, through sales, criticism and reviwes. The most recent of these can be found in [15].

5. Conclusions and further work

Although the presented algorithm allows to generate a model of the relationships between the combinations of parameter values and their associated perceptual category, its interpretability (although human-readable) depends on the quantity and quality of the rules. For example, on the variability that they present, it is not the same to have a set in a parameter with the values $\{0.2,0.4,0.6,0.8,1\}$, that with values $\{110.3, 1548\}$. So a higher level algorithm able to analyze the rules to generate a more compact model is desirable. Regarding the generalization process, although it suggests the user new combinations (for example in section 2.2 the rule extracted with d = 3 offers 18 combinations from 4 presets), compared to the dimension of the space these are few. So, more effi-

cient mechanisms of generalization are also desirable. For example, in [8] a mechanism that allows to generalize the sets that appear in the entries of the rules to the intervals between their maximum and minimum elements is presented. Future work also includes the extension of the rules, so that they cover larger regions of the space. A first version of these extensions is presented in [8], in which the sets of values found in the parameters of the rules are replaced for the intervals between the minimum and maximum values of the respective set. For example, the set $\{3,5,8\}$ is replaced by the interval [3-8]. The algorithm that realizes this task solves the contradictions that could be formed between the rules following a heuristic that looks for the set of rules covering the greater possible area. User test in which the users listened to material produced by these rules and classified it as belonging or not to the category of the rule is found in [8].

Current research focuses on start form the rules with intervals in their parameters, to build fuzzy rules that cover the entire space. For this, the intervals of the parameters are used as the cores of trapezoidal membership functions, whose supports are then adjusted by considering the other rules. In a similar way to the methodology presented in [16]. In this way, the observed space is fully classified. Finally, a deeper exploration of how the algorithm can be used as creative tools in the compositional process, in the sense in which [10] analyzes the genetic algorithms together with interactive evolution would yield interesting results.

References

[1] M. Schedel, (2011). "Oxford Handbook of Computer Music". Computer Music Jou rnal, 35(1), 105-107. 2011.

[2] A. McLean, R. Dean, (Eds.) "The Oxford Handbook of Algorithmic Music" Oxford University Press. 2018.

[3] S. Goldmann, "Presets digital shortcuts to soud" The Bookworm, an imprint of The Tapeworm, 2015.

[4] P. Dahlstedt, "Evolution in creative sound design". In Evolutionary computer music (pp. 79-99). Springer London. 2007.

[5] P. Dahlstedt, "Creating and exploring huge parameter spaces: Interactive evolution as a tool for sound generation." ICMC. 2001.

[6] N. Collins, "Interactive evolution of breakbeat cut sequences." Proceedings of Cybersonica, Institute of Contemporary Arts, London, England 2002.

[7] R. Dawkins, "The blind watchmaker: Why the evidence of evolution reveals a universe without design". WW Norton & Company. 1986.

[8] I. Paz, À. Nebot, F. Mugica, & E. Romero, "Modeling perceptual categories of parametric musical systems". Pattern Recognition Letters. 2017.

[9] I. Paz,À. Nebot, F. Mugica, & E. Romero "RULEX: a rule extraction algorithm for charting perceptual spaces" unpublished.

[10] P. Dahlstedt, "Thoughts on creative evolution: a meta-generative approach to composition." Contemporary Music Review 28.1 (2009): 43-55.

[11] J. Frnkranz, G. Dragan, L. Nada. "Foundations of rule learning". Springer Science & Business Media, 2012.

[12] C. Roads, "Sound composition with pulsars." Journal of the Audio Engineering Society 49.3. 134-147. 2001.

[13] Pearce, Marcus, D. Meredith, W. Geraint, "Motivations and methodologies for automation of the compositional process." Musicae Scientiae 6.2: 119-147. 2002.

[14] I. Paz "Visions of Space" https://bohemiandrips.bandcamp.com/album/visions-of-space 2017.

[15] Bandcamp daily "Meet the artists using coding, AI, and machine language to make music" https://daily.bandcamp.com/2018/01/25/music-ai-coding-algorithms/ 2018.

[16] M. Berthold, "Mixed fuzzy rule formation." International journal of approximate reasoning 32.2-3: 67-84. 2003.