Pattern Recognition Letters 105 (2018) 217–225

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Modeling perceptual categories of parametric musical systems

Iván Paz, Àngela Nebot*, Francisco Mugica, Enrique Romero

Computer Science Department BarcelonaTech, C/ Jordi Girona 1-3. Omega 005, Barcelona 08034, Spain

A R T I C L E I N F O

Article history:
Available online 11 July 2017

MSC:
41A05
41A10
65D05
65D17

Keywords:
Musical system
Modeling
Classification
Rule base system

A B S T R A C T

In computer music fields, such as algorithmic composition and live coding, the aural exploration of parameter combinations is the process through which systems' capabilities are learned and the material for different musical tasks is selected and classified. Despite its importance, few models of this process have been proposed. Here, a rule extraction algorithm is presented. It works with data obtained during a user auditory exploration of parameters, in which specific perceptual categories are searched. The extracted rules express complex, but general relationships, among parameter values and categories. Its formation is controlled by functions that govern the data grouping. These are given by the user through heuristic considerations. The rules are used to build two more general models: a set of "extended or Inference Rules" and a fuzzy classifier which allow the user to infer unheard combinations of parameters consistent with the preselected categories from the extended rules and between the limits of the explored parameter space, respectively. To evaluate the models, user tests were performed. The constructed models allow to reduce complexity in operating the systems, by providing a set of "presets" for different categories, and extend compositional capacities through the inferred combinations, alongside a structured representation of the information.

1. Introduction

Computer music is the application of computer technology in music composition [10], either to program computers that create music automatically, like algorithmic composition [20], or to help human composers to program music in real time, as is the case in live coding [6,16,28]. In these activities, parametric systems capable of generating different types of musical material are used [2,25]. Examples of parametric musical systems include sound synthesizers and signal processors, as well as melodic or rhythmic sequence generators. To develop a clearer idea of the wide range of possible parametrical musical systems, and therefore motivate the need for a methodology that creates a structured model of their capabilities, let us consider a couple of examples: In the first one, McCormack et al. [18] discuss the granular synthesis system "Chaos-Synth", developed by Miranda [19] for wind instruments. It consists of a bank of oscillators, whose individual frequencies and durations are controlled by the arithmetic mean of groups of cells taken from a cellular automata. In this case, the granular synthesis engine is parameterized by the cell states. The control parameters of the system, once the set of rules for the automata is written, are the cells, and the parameter space is the set of initial conditions with which the automata can be initialized. As a second example, consider a simple synthesis system based on two oscillators controlled by directly changing the values of their parameters. Suppose the system has two parameters: frequency 1 and frequency 2 both in Hz, with values ranging from a to b. In this case the system has an $[a, b]$ x $[a, b]$ space of possibilities, controlled by parameters frequency 1 and frequency 2. Even though these constructions can be framed in general structures (e.g, in additive or granular synthesis), they may have small changes or particular characteristics that modify their response. For example, they could have a different number of oscillators or envelope types, as well as different ranges in their parameters. Therefore, although there is a general idea of the behavior of the different architectures, it is necessary to test a new system to explore its possibilities.

By changing their parameters, the musician interacts with the parametric musical systems. Therefore, in all cases, the exploration of the different combinations of parameter values is the process through which the system's capabilities are learned, and the selection and classification of the musical material is performed. In [11] the commercial and cultural roles that specific parameter settings (called "presets") have had when they become sound "standards" is analyzed. Examples of presets include the classic configurations of an equalizer labeled as "Funk, Rock or Classical", digital plug-ins that mimic iconic instruments and new programs for the automatic audio mastering of a song depending on its genre. In a

* Corresponding author.
E-mail addresses: ivanpaz@cs.upc.edu (I. Paz), angela@cs.upc.edu (À. Nebot).

http://dx.doi.org/10.1016/j.patrec.2017.07.005
0167-8655/© 2017 Elsevier B.V. All rights reserved.

similar way, in algorithmic composition and live coding the selection of parameter configurations has to do with selecting settings that produce musical material with specific perceptual categories. Examples of this selection process are: given a system with two variable-frequency oscillators, to find all the frequency combinations that produce a consonant output (in acoustics a sound with harmonic partials), or a sound with "rough" quality. Although the exploration of parameter combinations in search of perceptual categories is a common activity in computer music, there have been few attempts to formalize the information produced during this process with the aim of using it for different musical tasks, such as the automatic creation of variations within a part of a piece while playing in front of an audience (live performance), or the execution of generative music algorithms [2].

Nonetheless, there are excellent examples of methods for finding sets of parameters that successfully produce entities with specific perceptual properties. For example, Collins [4], 5], Dahlstedt [8] applied interactive evolution [9], which uses human evaluation as the fitness function of a genetic algorithm for system parameter optimization. In [8], this technique was applied to sound synthesis and pattern generation tasks, while in [4,5], it was used for searching successful sets of arguments controlling algorithmic routines for audio cut procedures.

The present work is inspired by the methodologies of [5,8], and further focuses on structuring the information that is produced during the exploration process to build tools intended for the following objectives:

1. To provide a structured representation of how the values of the parameters are related with the user-selected perceptual categories.
2. To reduce the cognitive and operational complexity of the algorithms by providing a set of organized "presets" for the different preselected categories which can be used to automate some musical tasks, such as live coding.
3. To extend the compositional capacities by inferring new unheard (unexplored) parameter combinations consistent with the perceptual categories.

In this research, a methodology based on rules is developed for the modeling of perceptual properties. Our particular interest in a rule-based model relies on its interpretability. Rule models, in contrast with subsymbolic approaches (like neural net classifiers), are human readable information, which make them especially attractive for applications in the context of computer music. In addition, a rule-based model naturally functions as a set of presets, by encoding user associations (between parameter configurations and perceptual category). Furthermore, it is possible to extend this model to explore new regions of the space consistent with the assigned classes of the rules. An early version of the methodology can be found in [22].

The proposed methodology consists of four parts, shown in Fig. 1, together with their respective evaluation processes. The methodology has a data acquisition stage which is performed through a parametric exploration of a musical system, while searching for predefined perceptual categories (see # 1 in Fig. 1). During this process, the parameter combinations are labeled with their respective perceptual categories. Then, the data is grouped according to specific patterns, as explained in detail in Section 2. The resulting structures are called "Strict Rules" (see # 2 in Fig. 1). These rules are used to build a more general model, which extends the validity of the Strict Rules from points in the space to intervals. The result of this process, described in Section 3, is a set of "Inference Rules" (see # 3 in Fig. 1). Then, a next level generalization based on the set of Inference Rules is made through the design of a fuzzy classifier (# 4 in Fig. 1). This classifier covers the complete explored space and is presented in depth in Section 3.

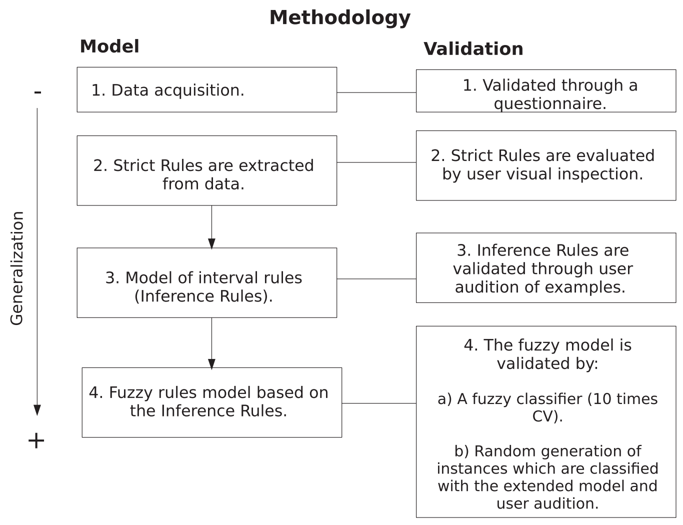
Methodology
Model
Validation
-
Generalization
+
1. Data acquisition.
2. Strict Rules are extracted from data.
3. Model of interval rules (Inference Rules).
4. Fuzzy rules model based on the Inference Rules.
1. Validated through a questionnaire.
2. Strict Rules are evaluated by user visual inspection.
3. Inference Rules are validated through user audition of examples.
4. The fuzzy model is validated by:
a) A fuzzy classifier (10 times CV).
b) Random generation of instances which are classified with the extended model and user audition.

Fig. 1. General methodology.

Sections 4 and 5 describe, respectively, the model evaluation process and the experiments performed to validate the different models. Section 5 also describes in detail the data acquisition process and the user interface used for the parametric exploration. Finally, Sections 6 and 7 present, in turn, the discussion followed by the conclusions and further work.

2. Strict Rules extraction

2.1. Algorithm design considerations

The Strict Rules extraction algorithm was designed to identify complex but general relationships among the values in the system parameters and the perceptual categories assigned by the user. The idea is to reorder the data to make visible specific patterns. In this way, it produces a "structured" and interpretable representation of the input data. In Section 3, this representation is used to infer new combinations of parameters consistent with any of the user-defined perceptual categories. The algorithm was inspired by a rule extraction algorithm developed by Castro et al. [3]. However, it has two fundamental differences. First, the result does not depend on the order of the data. Second, the algorithm can create rules requiring fewer conditions. These are discussed in Section 2.3.

The input data are organized as data pairs (parameter values, perceptual category), which are seen as deterministic input-output relations. In these data, the algorithm searches iteratively for patterns. Specifically, it looks for combinations of parameters with the same category that differ only in one parameter value (let us say p_j). In that case, if the absolute difference of the p_j values that differ is less than a certain threshold (t), the combinations are grouped together into one structure. This structure has the same values of the grouped combinations in all its parameters, the same category and a *set* in p_j that contains the values that differ. Before presenting the algorithm let us define the function that calculates the thresholds.

2.1.1. Threshold function

The thresholds are calculated through functions that are declared in advance by the user. These are based on heuristic considerations, i.e. on a priori knowledge of the variables that the parameters represent. For example, the frequency or the number of upper harmonics added to a signal. Some ideas on how to automate this process are presented in Section 7. For each parameter p_j a threshold function $t_{p_j}(x)$ is declared. It assigns a threshold to

Table 1
Consider the parameter combinations X_1, X_2 and X_3 all with class (or perceptual category) *rough* and differing only in the *j*th parameter p_j. i.e., $x_{1,k} = x_{2,k} = x_{3,k} \,\forall\, k \neq j$. Suppose that $x_{1,j} = 7$, $x_{2,j} = 11$, $x_{3,j} = 25$. The result of grouping this combinations using the Treshold function of Eq. (1) and combining the respective thresholds using $g(t_{p_j}(x_1), t_{p_j}(x_2)) = Min(t_{p_j}(x_1), t_{p_j}(x_2))$ are shown at the bottom of the Table. Note that in $RuleX_1X_2$ entrance $x_{1-2,k} = x_{1,k} = x_{2,k} \,\forall\, k$.

Combination	p_1	...	p_j	...	p_N	Class
X_1	$x_{1,1}$	...	$x_{1,j} = 7$	...	$x_{1,N}$	rough
X_2	$x_{2,1}$	...	$x_{2,j} = 11$	...	$x_{2,N}$	rough
X_3	$x_{3,1}$	...	$x_{3,j} = 25$	...	$x_{3,N}$	rough
Result						
$RuleX_1X_2$	$x_{1-2,1}$	...	{7, 11}	...	$x_{1-2,N}$	rough
X_3	$x_{3,1}$	...	$x_{3,j} = 25$	...	$x_{3,N}$	rough

each value *x* of the parameter p_j. Given x_1, x_2, they will be grouped in a *set* if Eq. (1) is satisfied.

$$|\, x_1 - x_2 \,| < g(t_{p_j}(x_1), t_{p_j}(x_2)) \tag{1}$$

g is a function of two variables, that can be defined as $Max(t_{p_j}(x_1), t_{p_j}(x_2))$ or $Min(t_{p_j}(x_1), t_{p_j}(x_2))$ or a constant function, for example $g(t_{p_j}(x_1), t_{p_j}(x_2)) = inf \,\forall x$, etc. Functions *t* and *g* control, for each parameter, how values are grouped into sets.

Example: Let p_j be a parameter that controls a frequency (Hz). One possible definition of the function $t_{p_j}(x)$ could be as in Eq. (2).

$$t_{p_j}(x) = \begin{cases} 10 & \text{if } x \le 20\,Hz \\ 80 & \text{if } x > 20\,Hz \end{cases} \tag{2}$$

Suppose that we have the parameter combinations X_1, X_2 and X_3, shown at the top of Table 1. They differ only in the value of parameter p_j, i.e., $x_{1,k} = x_{2,k} = x_{3,k} \,\forall\, k \neq j$. Suppose also that $x_{1,j} = 7$, $x_{2,j} = 11$, $x_{3,j} = 25$. The bottom of Table 1 shows the result of grouping this combinations using the Threshold function of Eq. (2) and combining the thresholds using $g(t_{p_j}(x_1), t_{p_j}(x_2)) = Min(t_{p_j}(x_1), t_{p_j}(x_2))$. For the first two values, $x_{1,j} = 7$ and $x_{2,j} = 11$, we have $t_{p_j}(7) = 10$ and $t_{p_j}(11) = 10$. For the values 11 and 25, $t_{p_j}(11) = 10$ and $t_{p_j}(25) = 80$. As $|7 - 11| < Min(10, 10)$ and $|11 - 25| \not< Min(10, 80)$ the combinations X_1 and X_2 are grouped into $RuleX_1X_2$ and X_2 is not grouped with X_3, so X_3 remains alone.

The thresholds allow the user to define how "close" two values can be in order to be placed in the same rule. For example, suppose that for a specific parameter (p_j), the threshold is set equal to infinity for all of their values. I.e., $t_{p_j}(x) = inf \,\forall\, x$ and $g(t_{p_j}(x_1), t_{p_j}(x_2)) = inf$. Then, all parameter combinations in the data, differing only in the value of that parameter will be grouped into a single rule no matter how "separated" they are. This could produce great variability in the combinations described by the rule. In contrast, suppose that a threshold $t(x) = constant < inf$ is set for all the values. The result would then be a set of rules, each of them containing values that, when ordered from minimum to maximum, will not be separated from each other by more than that constant.

2.1.2. Considerations on thresholds applications

Thresholds determine, for each parameter, the maximum distance between two adjacent grouped values, and can be used for different purposes. For example, to create small transitions when using the rule instances that have been grouped with a small threshold, or big transitions (to create contrast) in the opposite case. For a discussion of how stepwise perceptual transitions (or small perceptual changes in the sound) have played an essential role in music composition, from Gregorian chant to 20th century music, see [7]. Furthermore, structures to visualize relationships between parameters and perceptual categories, as the rules can be thought of, can also be interesting from an analytical point of view. For example, they can be used for computational modeling of musical styles [23].

2.2. Algorithm inputs

The inputs for the algorithm are:

1. **data**: {$\mathbf{X}_1$, $\mathbf{X}_2$, ..., $\mathbf{X}_m$}, where each $\mathbf{X}_i = ((x_{i1}, x_{i2}, ..., x_{in}), y_i)$, x_{ij} denotes the value of the *j*th parameter of the *i*th combination explored, and y_i denotes the perceptual category. Index *i* satisfies that $1 \le i \le m$ and *n* is the number of parameters.
2. **thresholds**: Array containing, for each parameter, a function $t_{p_j}(x)$ that calculates the threshold for each value, and the function *g* (defined in Section 2.1.1) to combine the thresholds.

2.3. Algorithm

The algorithm is currently implemented in the SuperCollider programming language [17,29] since it can also be used as a synthesis engine to implement the sound generators. As mentioned, the algorithm searches, parameter by parameter, instances differing only in one parameter value and belonging to the same perceptual category. Given that the result of such search depends on the order of parameters, the algorithm starts by calculating all the possible parameter permutations of the data. Then it applies the search to each permutation and the independent results are aggregated by eliminating the redundant and repeated rules at the end. This is a difference with the algorithm proposed in [3] that searches using only the initial order of the data.

The search is performed as follows:

1. For each column index the corresponding column is excluded from the data and identical rows are searched. Two rows are identical if they are equal entry-to-entry. If the entries are *"sets"*, they are considered identical if they contain the same elements.
2. Then, the values of the excluded column for the selected (identical) rows are taken.
3. These values are sorted from min to max and used to form subsets. Each subset contains all the subsequent values whose absolute difference is less or equal to their corresponding threshold (distance).
4. Finally, the algorithm creates one rule for each created subset. The rules have the values of the selected rows in all the parameters, and a set with all the values of the subset in the index of the excluded column. The rows used to create the rules are eliminated from the data and the rules are added.

As an example of set and rule creation suppose a constant threshold of 0.2 for parameter *k*. If we have values = {0.1, 0.3, 0.5, 0.8, 0.9, 1.1}, we will have the sets: {0.1, 0.3, 0.5} and {0.8, 0.9, 1.1}. As we have excluded column *k*, the new rules created will be : $x_1, x_2, ..., x_{k-1}, \{0.1, 0.3, 0.5\}, x_{k+1}, ...x_n, y_k$, and $x_1, x_2, ..., x_{k-1}, \{0.8, 0.9, 1.1\}, x_{k+1}, ...x_n, y_k$. As the process of creating all the permutations is computationally expensive, the user can decide the order of the parameters and avoid this step. Note that, through the process of constructing and separating sets, the Strict Rules algorithm allows the grouping of rules starting from sets containing only two instances. Unlike [3] algorithm in which, for a rule to be formed, it is required that the set contains all the possible values that the analyzed parameter can take. Therefore in one parameter we may have different rules covering different ranges of values. This gives the algorithm the necessary flexibility to find regularities with different levels of generality.

2.3.1. Algorithm pseudocode

```
1. permutations = array with the possible permutations of the parameter indexes (excluding the output)
```

Table 2
Extension of a Strict rule into an Inference rule. The list of values in parameter p_j is extended to the interval between the minimum (a) and maximum (b) values.

Rule Type	p_1	...	p_j	...	p_n	Class
Strict Rule	$x_{k,1}$	...	$\{a, b, c, d\}$	...	$x_{k,N}$	rough
Inference Rule	$x_{k,1}$	...	$[a-d]$	...	$x_{k,N}$	rough

```
2. For each permutation:
   temporal_data = copy of data
   For each index j in permutation (j from 0 to # parameters):
      Exclude column permutation[ j ] from temporal_data
      For each row (in temporal_data):
         a. Look for identical rows
         b. Collect from the identical rows the values located at the
            excluded column
         c. Create sets and new rules with the collected values (See
            Section 2.3.2)
         d. Eliminate the identical rows
         e. Add the new rules to temporal_data
      Add temporal_data to sets_of_rules
3. Eliminate redundant and repeated rules from sets_of_rules
4. Return sets_of_rules (array to store the result of the algorithm ap-
   plied to each permutation)
```

2.3.2. Create sets and rules

Create sets

To create the new sets proceed as follows:

1. *Sort set (array)*
2. *Split the set in subsets S_i such that $\forall x_i, x_{i+1} \in S_i \mid x_i - x_{i+1} \mid \leq t^*$ Where t^* is the assigned threshold for elements x_i, x_{i+1} by the functions g, t (described in Section 2.1.1) of parameter j (located at **thresholds**[j]).*

Create rules

To create the new rules proceed as follows:

1. *Take the current row (set of parameters) used to look for identical rows.*
2. *For each of the sets created in the previous step (**Create sets**) build a new rule by replacing the excluded column (j) by the selected set.*

3. Inference Rules and Fuzzy Rules model

3.1. Construction of the Inference Rules

As mentioned, the Strict Rules do not describe unheard parameter combinations. They only structure the data by grouping the regularities found. This allows us to separately perform the process of structuring the existing information and the process of building a model able to infer new, unheard combinations, consistent with the preselected perceptual categories. To do this, a set of Inference Rules is created. These rules are extensions of the Strict Rules created with the algorithm described in Section 2. To extend the rules, the values of the *sets* contained in their parameters are replaced by the intervals between the minimum and maximum values for each *set*. For example, let us suppose that we have the rule shown in Table 2, containing values $\{a, b, c, d\}$ in parameter p_j. Without the extension, this rule only describes the cases formed with the combinations of the values in the *set* and the values of the other parameters (which could also be a *set*). To build the inference rule, we substitute the *set* for the interval $[a-d]$, a and d being the minimum and maximum values of the *set*, respectively. The extended rule comprises the combinations of the parameter values with all the values in the interval.

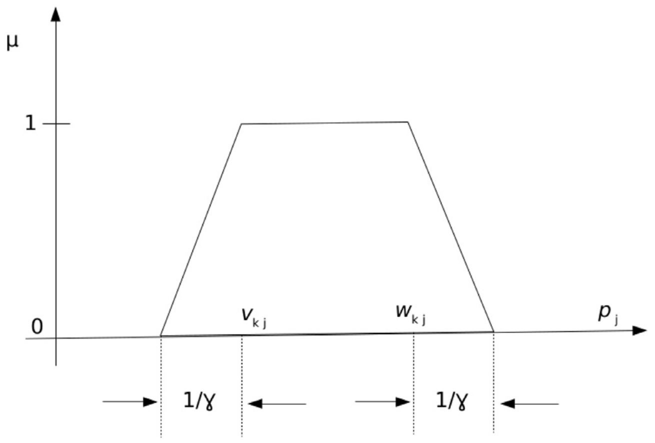

Fig. 2. Membership function for the classifier. The values $v_{k,j}$ and $w_{k,j}$ are, respectively, the minimum and maximum values of parameter j in rule k. The parameter γ controls the "slopes".

3.2. A fuzzy classifier based on Inference Rules

To extend the model to cover all the space within the explored limits, a fuzzy if-then classifier, based on the Inference Rules, was designed. It was built as follows:

First, consider the trapezoidal membership function of Eq. (3), which is represented in Fig. 2.

$$\mu_{k,j}(p_j) = \begin{array}{l}\frac{1}{2}[max(0, 1 - max(0, \gamma * min(1, p_j - w_{k,j}))) \\ +max(0, 1 - max(0, \gamma * min(1, v_{k,j} - p_j)))]\end{array} \tag{3}$$

In this function [14] the index k represents the rule number and the index j counts the parameters. The values $v_{k,j}$ and $w_{k,j}$ are, respectively, the minimum and maximum values with membership $\mu(x) = 1$ of the kth rule jth parameter. The parameter γ controls the "slopes" of the trapezoid and is calculated so that the slopes intersect the x-axis at the minimum and maximum values of the parameter.

To build the classifier, all features are scaled into [0, 1]. Then, for each rule R_k, a trapezoidal membership function is assigned to each parameter p_j in the following way:

1. If the parameter contains an interval, let us say $[a-b]$, then $v_{k,j}$ = a and $w_{k,j}$ = b. In this case, the trapezoidal membership function has maximum membership values between the extremes of the interval.
2. If the parameter contains a single value, then $v_{k,j} = w_{k,j}$. Thus, we obtain a triangular membership function centered at that value.

To classify a new parameter combination, the classifier operates as follows: Let P be the combination sent to the classifier. P is a combination of p_j parameter values. Then, for each rule R_k, it calculates the membership values $\mu_{k,j}(p_j)$ for each parameter p_j. The firing strength $\tau_k(P)$ [14] of a rule R_k, which measures the degree to which the rule matches the input parameters, is defined as the minimum of all the membership values obtained for the parameters (see Eq. (4)), i.e:

$$\tau_k(P) = min\{\mu_{k,j}(p_j)\} \tag{4}$$

Once the firing strength has been calculated for all rules, the assigned class (i.e. category; Eq. (5)) will be equal to the class of the rule with maximum firing strength.

$$Class(P) = \text{Class of } R_c \text{ where } C = \underset{k}{argmax}\{\tau_k(P)\} \tag{5}$$

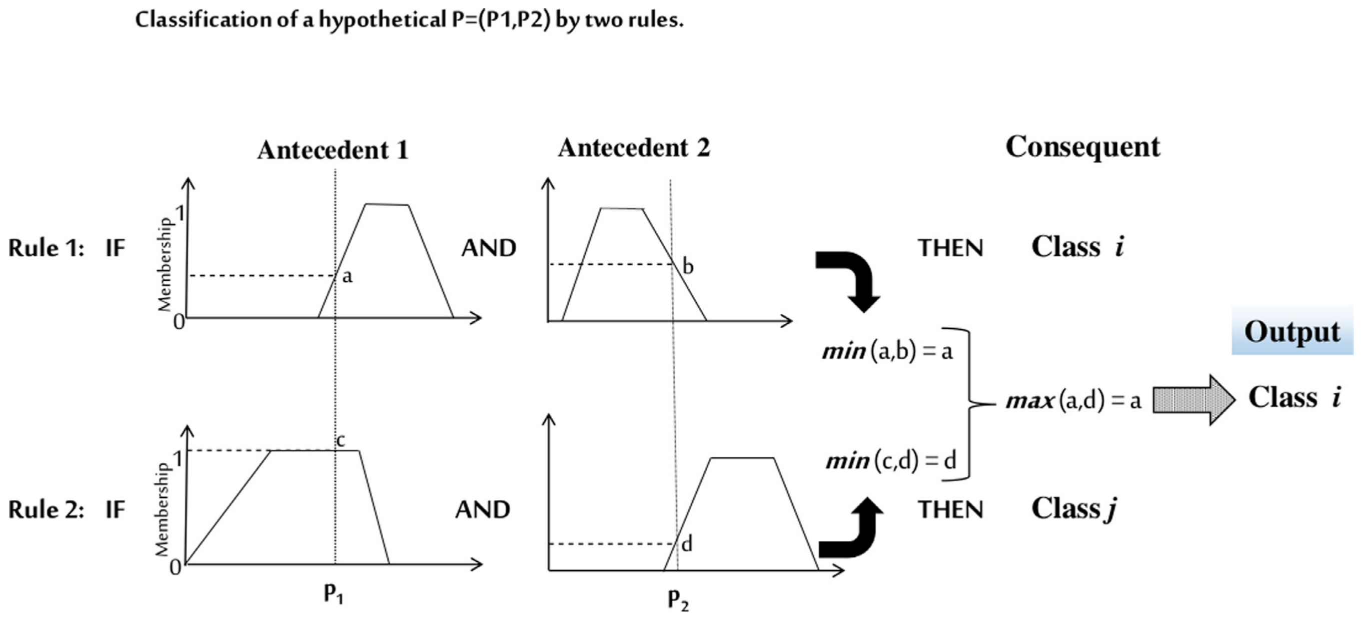

Fig. 3. Example of the classification process for a system of two rules with two parameters p_1 and p_2. A new combination $P = (p_1, p_2)$ is sent to the classifier. For the first rule $\mu(p_1) = a$ and $\mu(p_2) = b$. The minimum of these values is a. In the case of the second rule $\mu(p_1) = c$, $\mu(p_2) = d$ and $min(c, d) = d$. Finally, the $max(a, d) = a$ and therefore the class assigned to the instance is *Class i*, which is the class associated with the first rule.

An example of the classification process for a hypothetical system with only two rules and two parameters is shown in Fig. 3. The parameter combination $P = (p_1, p_2)$ is sent to the classifier. To classify this combination, for each rule, the degree of membership of each p_i given by the i^{th} membership function of the rule is calculated and the minimum of these values is taken. Then, a set containing the minimum value of each rule is formed. Finally, the output of the system is the class associated with the rule corresponding to the maximum value of that set. If more than one rule has the same firing strength, a random decision is taken.

4. Model evaluation

The model evaluation was based on the ideas suggested by Pearce et al. [23]. They discuss specific motivations for the development of programs for composing music, suggesting specific criteria for its evaluation depending on the case. They identify four activities: algorithmic composition, the design of compositional tools, the computational modeling of musical styles and the computational modeling of music cognition. Although the algorithms presented here were originally designed for personal compositional purposes and therefore could be cataloged within the algorithmic composition activity, the further development of the system was designed to be a compositional tool, i.e. it is a tool intended to be used by composers other than the designers. As proposed by Pearce et al. [23], we therefore performed evaluation tests to ensure that the behavior of the software satisfies the requirements (unit tests), as well as to examine its performance in different scenarios (application tests).

The different models presented were evaluated as follows: The Strict Rules were validated in experiments through user visual inspection, that is, the extracted rules were grouped in classes and shown to the user for its validation (in question 2.1 of the Evaluation questionnaire below). The Inference Rules were evaluated through user testing. In this case, the users evaluated, whether or not, random combinations produced from the generalized intervals were consistent with the preselected perceptual categories. To evaluate the Fuzzy Rules model, users evaluated random combinations covering the whole range of parameter values. The evaluation of the users was compared with the results of the classifier described in Section 4.2. Finally, the accuracy of the classifier was calculated using the latter data and the standard technique 10-fold cross-validation. The details are presented below.

4.1. Evaluation questionnaire

The tests results were registered through a questionnaire. As the system has different components that relate with the user in different ways (user interface, extracted rules and generalization stage) these were addressed in separate sections. The questionnaire is shown below.

4.1.1. Assessment questionnarie

The data and threshold functions were saved for each user.

1. Data acquisition process
   (a) Interface evaluation: Do you consider that the interface is intuitive and suitable for exploring the space of parameter combinations?
   (b) Instructions evaluation: Do you feel the instructions given by the interviewer are an efficient way to explore changes in the parameters?
2. Extracted rules
   (a) Do the rules give you interpretable information about the relationship of the parameter values with the perceptual category of the output?
3. Generalization
   (a) Consistency: Do you perceive the produced parameter combination to be consistent with the selected category?
   (b) Novelty: Do you find novelty in the unheard patterns?

To answer the questions, we adopted a user-focused evaluation through Likert scale feedback [15] as proposed by Wanderley and Orio [27]. With the following categories: strongly agree, agree, neutral, disagree, and strongly disagree. We used this approach to evaluate the data acquisition process (interface and parameter exploration methodology), the interpretability of the extracted rules, and the novelty of the unheard parameter combinations created in the generalization step. To evaluate if the category of the new

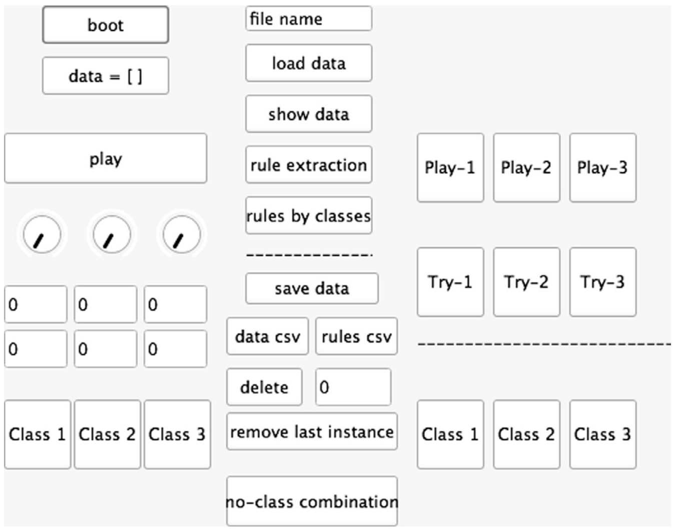

Fig. 4. Interface used to explore the parametric space and to collect data. The left side allows to boot the system, initialize the data, tweak the parameters and classify the combinations into three categories. The central part performs the rule extraction process. The right side allows the user to listen the Strict Rules (Play buttons) or the Inference Rules (Try buttons), as well as to reclassify the instances (Class buttons).

unheard combinations perceived by the user was consistent with the selected category (question 3.1), a consistency score was used, similar to the musical Turing Test presented in [26]. The general idea is that, in computer music, we may desire the systems to behave in a human-like fashion. In our case, the produced examples should emulate patterns that would be selected by the human user as belonging to a particular category. Therefore, as Stowell et al. [26] pointed out, “the degree of observed confusion between human and automated response is an appropriate route for evaluating systems which perform human-like tasks”. In the tests, the Inference Rules generated new parameter combinations, and the users evaluated whether they were consistent, or not, with the requested category. If so, we had a success and otherwise a failure.

4.2. Evaluation of the fuzzy model

Two tests were performed to evaluate the fuzzy model. First, parameter combinations (from the entire observed space) were produced, and the classifications of the model and the user were compared. Second, we used the classifier described in the Section 3.2 to estimate the model accuracy. A 10-times cross-validation standard technique was used on the data acquired during these experiments.

5. Experiments

The experiments were conducted with students of the Real Time Interaction Class of the Master’s Degree in Sound and Music Computing of the Pompeu Fabra University (winter term of 2016–2017), as well as with computer music composers and audio technology developers. The systems and the collected data are available at [21]. Individual sessions were held in which the surveyed were introduced to the system interface and functionalities. After that, the participants were given time to become familiarized with the system before beginning the experiment.

For the user-tests, an analytic interface (shown in Fig. 4) as described by Hunt and Kirk [13] was designed. It allowed the users to listen, tweak and perceptually categorize (into three classes given the perceptual categories of the systems described in Section 5.2) the parameter combinations while seeing their values in a display. It also allowed the captured data to be reviewed. Once the data were collected, the user executed the rule extraction algorithm and listened to parameter combinations in the following modalities:

1. Patterns contained in the generated Strict Rules, such that they had already been heard and categorized by the user.
2. Patterns inferred through the Inference Rules, i.e, assumed to be within some predefined perceptual category, but which had not been heard.

Users were asked to collect at least thirty categorized combinations by following the instructions shown in Section 5.1. After this, the rule extraction process was performed and the obtained Strict Rules were inspected by the users. Finally, twenty unheard combinations were produced with the Inference Rules. These were evaluated as being consistent or not with the categories that they described, as well as whether or not they showed novelty.

5.1. Instructions for exploring the parametric space

In order to favor rule formation and to avoid having many instances not grouped in any rule, users were provided with specific instructions to vary the parameters while exploring the space. These instructions allow to explore the space in an ordered way, so that changes occur in one parameter at a time. Participants were asked to explore the space as follows:

1. Without restrictions, vary the system parameters until a new point is found within the desired perceptual category. Save the parameter values with its associated category.
2. From that point, tweak one parameter at a time while keeping the others constant. If the new combinations belong to any of the desired categories save them with its respective label. In this way, changes in one or more parameters can be explored.
3. When enough has been explored around the combination, freely vary the parameters until a new point with a desired perceptual category is found.

5.2. Sound generators

For the experiments, two different sound generators were selected. The first system, was a single band limited impulse oscillator with well defined acoustic properties of its different possibilities (Section 5.2.1). The second was a slightly more complex system for additive synthesis for which an arbitrary perceptual property was chosen (Section 5.2.2). The first system, was selected because of the great amount of psychoacoustics documentation describing the human perception of their different aural possibilities, allowing the validation of the obtained results (rules). Moreover, the simplicity of the system made it an excellent candidate for the interviewees to become familiar with the methodology. The second system was chosen given that consonance is a well defined perceptual property, and considering that, as additive synthesis is generally the first synthesis technique to be studied, the users were also familiar with this system.

5.2.1. Single band limited impulse oscillator

The single band limited impulse oscillator “Blip”, produces a fundamental frequency to which a certain number of upper harmonics, all with equal amplitude, are added [1]. It is available as a unit generator (UGen) in SuperCollider. It has three parameters: *freq*, the fundamental frequency; *numharm* which controls the number of upper harmonics added; and *amp*, the output’s amplitude. The parameters can range in the following way: the fundamental frequency can take values from 0Hz, with no audible result, to 20 KHz which is the upper audible limit. However, as the audition capacity is reduced with age and environmental factors, the upper limit is sometimes considered around 15 KHz. The number of added harmonics range from 0, which leads to a pure tone frequency of the chosen fundamental, to any (theoretical) number of harmonics. However, considering the aliasing processes, this number is limited by the sample rate and the chosen fundamental frequency. Finally, the signal amplitude is normalized between 0 and 1.

Configuration for the experiments

For the experiments, the values were restricted within the following ranges. The frequency took values in the interval [0Hz, 400Hz], the number of upper harmonics in [0, 100], and the amplitude in [0, 1]. This is a bounded space where the perceptual properties (described next) are clearly expressed. Although this is a simple generator, the different aural possibilities produced by its parameter combinations clearly define distinct perceptual subspaces. For the experiment, we worked with three perceptual properties. These are outputs perceived as **rhythmic, rough**, and **pure-tone**. For further reference of the Blip generator, the reader is referred to [24].

Generally speaking, outputs perceived as rhythmic are characterized by low fundamental frequencies (less than 15 Hz). The number of upper harmonics controls the pitch of the produced pulse (or beat). A low number of harmonics produce constant beats with low pitch, while a higher number of upper harmonics produce the same constant beating, but with higher pitch. Outputs perceived as rough have fundamental frequency values between 15 and 35 Hz. The number of upper harmonics plays a similar role as in the case of rhythmic outputs. Finally, outputs perceived as pure-tones are produced by any frequency greater that 20 Hz without upper harmonics. However, combinations of low-mid to high frequencies (e.g greater than 80 Hz) with medium to high number of upper harmonics (for example, between 40 and 50) can sometimes be perceived as pure-tones. This phenomenon is accentuated due to the effects of comparison on the perception. For example, when hearing something highly rough and afterwards hearing a slightly rough combination, the latter can be perceived more like a pure tone. Also, it is worth saying that the regions described are not crisp, and the perception may vary from one user to another, as well as with the mentioned perception by comparison effect. For the experiments, $g(t_{p_j}(x_1), t_{p_j}(x_2)) = t_{p_j}(x_1)\ \forall\ p_j$. The threshold functions for the parameters were set, for all x, as in Eq. (6).

$$t_{p_j}(x) = \begin{cases} inf & \text{for any freq} \\ inf & \text{for any numharm} \\ 1 & \text{for any amp} \end{cases} \tag{6}$$

5.2.2. Sawtooth wave additive synthesis

As a second sound generator, a simple Additive Sawtooth Wave Synthesizer was selected. It consisted of four sawthooth waves with frequencies *freq1, freq1 - 1, freq2, freq2 + 1* measured in Hz, with a general normalized amplitude, *amp*. The system parameters were: *freq1, freq2* and *amp*. The system output is the sum of the four frequencies times the amplitude. This synthesizer was also implemented in SuperCollider.

Configuration for the experiments

For the experiments, the frequencies took values in the interval [0Hz,300Hz]. The sought perceptual property was defined as follows: The frequency space was divided in ranges of 100 Hz. At the first range [0Hz, 00Hz] *freq1* was set to 50 Hz. Then, *freq2* was tweaked and all the outputs that exhibited **consonance** between *freq1* and *freq2* were selected at the discretion of the user. For example, frequencies 50 Hz and 100 Hz given that one is the octave of the other. For ranges [100Hz, 200Hz] and [200Hz,300Hz] *freq1* was set equal to 200 Hz and 300 Hz respectively, and values of *freq2* were searched in the same way. *Consonant* values were freely chosen by the users. They were able to select, for example, only 50 and 100 Hz at the first interval, or 50, 100 and 75 Hz (the perfect fifth of 50), and so on. These values could be chosen guided simply by audition or analytically by using the interface. For the experiments $g(t_{p_j}(x_1), t_{p_j}(x_2)) = t_{p_j}(x_1)\ \forall\ p_j$ and the thresholds were defined as in Eq. (7).

$$t_{p_j}(x) = \begin{cases} 10 & \text{for any freq1} \\ 10 & \text{for any freq2} \\ 1 & \text{for any amp} \end{cases} \tag{7}$$

Remember that consonance can be defined, acoustically, considering the coincidence of partials [12] and the combined spectral distribution of the sound. There are also subjective variations dependent on the cultural context. Nevertheless, in this case, each subject evaluates its own material and therefore only changes in perception produced by the memory of past events (as the ear works by comparison) influence perception.

5.3. Results of the experiments

Ten subjects were interviewed, and all tested both generators. The evaluation of the systems was done through the presented questionnaire. The results evaluating the Strict and Inference Rules are presented in Table 3. Table 4 presents examples of Strict Rules for the Blip system. And Table 5 presents the evaluation of the fuzzy model. This includes the accuracy of the classifiers and the coincidence degree of the classes assigned to the new combinations by the users and the classifier.

5.3.1. Evaluation of the Strict and Inference Rules

Table 3 contains the results of the questionnaire evaluating the Strict and Inference Rules models, for the experiments carried out with both generators. The numbers below the five quantifiers of the Likert scale indicate the number of users who selected that option. In the case of the Inference Rules consistency (column 2, rows 8 and 9), the numbers after "Consistent" and "Inconsistent" express the average of the percentages calculated for the twenty produced combinations. An example of the Strict Rules obtained during the experiments is shown in Table 4. The second row shows a rule for the **Rhythmic** category. It should be read in the following way: If Frequency is 2.3 OR 3.94 AND Number of upper harmonics is 87.9 AND Amplitude is 4.7 THEN the combination is categorized as rhythmic. As can be seen, the rules obtained and shown in Table 4 coincide with the ranges for the perceptual properties described in Section 5.2.1. For example, for the output to be perceived as **Tone** the Number of upper harmonics should be low, or zero in this case.

5.3.2. Evaluation of the fuzzy model

The results of the experiments evaluating the fuzzy model are presented in Table 5. The second row contains the mean accuracy and standard deviation for the classifiers built for each data set generated by the users during the tests. The accuracy was calculated by using 10-fold cross-validation. The third row shows the mean percentage of coincidence between the categories assigned by the user and the classifier to new unheard combinations. Such combinations were taken randomly from the whole considered parametric space.

6. Discussion

For the purposes of this work, we were mostly interested in the evaluation of the perception of the Strict Rules in terms of interpretability, and the ability of the Inference Rules to produce unheard combinations consistent with the categories and the novelty

Table 3
Synthesis of the questionnaire results for the Blip and Sawtooth generators. The numbers below the quantifiers of the Likert scale denote the number of users that selected such option. The percentages in 3.1, are the average percentages of the numbers assigned by the users to the consistency of the twenty combinations produced by the Inference Rules.

System	Question	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
Blip	1.1 Interface evaluation	2	5	2	1	–
Sawtooth	1.1 Interface evaluation	2	5	2	1	–
Blip	1.2 Instructions	3	5	2	–	–
Sawtooth	1.2 Instructions	3	5	2	–	–
Blip	2 Strict Rules evaluation	6	3	1	–	–
Sawtooth	2 Strict Rules evaluation	7	2	1	–	–
Blip	3.1 Consistency of Inference Rules combinations	Consistent: Inconsistent:		90% 10%		
Sawtooth	3.1 Consistency of Inference Rules combination	Consistent: Inconsistent:		85% 15%		
Blip	3.2 Novelty	–	4	4	2	–
Sawtooth	3.2 Novelty	2	6	2	–	–

Table 4
Example of the Strict Rules obtained during the experiments for the Blip generator.

Category	Frequency	Number of harmonics	Amplitude
Rhythmic	2.3 OR 3.94	87.9	4.7
Rough	14.8	22 OR 47,9 OR 100	0.35
Pure-Tone	218.5 OR 323.7 OR 386.7 OR 400	0	0.12

Table 5
The second row contains the mean accuracy ± standard deviation for the classifiers build with the data collected during the experiments using 10 fold cross-validation. For each user, a classifier was build and tested. The third row shows the mean percentage of coincidence between the classifier and the user, in the classification of new combinations of parameters (taken over all parameter input space).

System	Blip	Sawtooth
Mean accuracy ± Standard deviation	0.955 ±0.0008	0.930 ± 0.0024
Coincidence user and classifier	93%	88%

of these combinations. The evaluation of the interface and the instructions for exploring the space (questionnaires 1.1 and 1.2), let us know that the collected data actually has the desired structure. That is: a combination of parameters associated with a particular perceptual category. If we see the numbers from second to fifth rows of Table 3, we can say that both the interface and the instructions to explore the space were considered appropriate for the data collection.

The results evaluating the Strict Rules, sixth and seventh rows of Table 3, show that the structure with which the rules are presented, effectively allowed the user to have an image of the value regions (within the parameters) that "codify" for a certain category. In many cases, the visualization of the rules allowed the users to verbalize some general ideas to describe its own behavior.

The eighth and ninth rows of Table 3, that describe the consistency of the Inference Rules, show that, in the case of the analyzed systems, it is possible to use the proposed generalization to extend the grouped values (sets) to regions maintaining their perceptual properties. In the case of the second generator, the consistency values are slightly lower, but the combinations are perceived with novelty. However, looking at the novelty evaluation of the Blip, the produced combinations exhibit moderate novelty. This is due in part to the homogeneity in the perception of this system. One way to overcome this would be to produce more risky combinations, for example by extending the cover regions by grouping adjacent intervals. However, when informal tests were performed, it was clear that, by doing this, there is a higher risk of leaving the perceptual categories. So, it is necessary to define a criterion for when the adjacent intervals can be grouped together.

This problem is directly related to the choice of the thresholds, that at the moment it is the user's complete responsibility. During the experiments it was clear that the consistency associated with the Inference Rules depends on the thresholds used. Since this determines how "separated" the numbers within the *sets* are, and therefore the "length" of the generalization interval. In the experiments the thresholds were initially set by heuristic considerations and then used to process all users' data. Therefore, developing a way to obtain the appropriate thresholds from the data is important for the quality of the Inference Rules, and to free the user from the responsibility of tuning the thresholds. In the next section we discuss work in progress to automate this process.

Table 5 presents the results that compare the category assigned to new combinations by the user and the Fuzzy Rules model (third row). It also presents the accuracy of the classifiers obtained with the standard 10-fold cross-validation (second row). Given that the results are similar between them, this allows us to have a positive evaluation of the Inference Rules model extended to Fuzzy Rules covering the whole space.

7. Conclusions and further work

In this paper, we present a methodology for modeling perceptual categories in parametric musical systems. It begins by grouping the variations that can occur in the value of a parameter without changing the perceptual category associated with the system output. The result of this analysis is an interpretable structure (Strict Rules) that was validated through user tests. Based on the Strict Rules, we constructed the first generalization. The new model extends the coverage of each Strict Rule. For this, the sets of values contained in their parameters are extended to intervals between the minimum and maximum values of each set. The resulting rules are called Inference Rules. These were validated by generating random combinations from the rules, which were evaluated by users as consistent or not with the expected perceptual category.

A second generalization was then constructed, covering the whole space by associating a trapezoidal membership function with each entry of the Inference Rules. For a new entrance, the output of the model was calculated by using the rules in a fuzzy classifier. This model was validated by generating new parameter combinations of the whole considered space and then, by comparing, for each combination, the categories assigned by the users and the model. In addition to this, the accuracy of the classifiers was calculated by using the 10-fold cross-validation method.

The presented models allow to codify the existing relationships between the parameter values combinations of the generative systems, and the perceptual categories assigned. However, the effectiveness of the model, highly depends on the selection of functions for calculating the thresholds. In this case, this responsibility lies on the user, so it is necessary to be familiar with the systems in order to choose the thresholds properly. Our current research focuses on a way to automatically find these functions. A possible solution would be to use the information given by the data for the grouping of the values. For example, once we have put together all the combinations that differ only in one parameter, and the set containing the different values has been formed, we split the set into subsets by grouping the values regardless of how distant they are. This is possible as long as there is no value between them that shares values in the other parameters, but has a different perceptual category associated. If this is the case, we group the values in two subsets to the “left” and “right” of that value. By doing this, the grouped rules will be naturally split into subsets without containing examples with a different label. This would eliminate the need for thresholds for data grouping, leaving them for more creative processes.

Acknowledgments

The kind help of Angel Faraldo and the Befaco collective for the realization of the tests must be acknowledged. This work was supported by the National Council of Science and Technology of Mexico (CONACyT) 240514. We also thank the anonymous reviewers whose wise feedback has been essential for improving the reading of this contribution.

References

[1] Blip, Blip, 2016, (http://doc.sccode.org/Classes/Blip.html). Accessed: 2017-01-25.
[2] A.R. Brown, A. Sorensen, Interacting with generative music through live coding, Contemp. Music Rev. 28 (1) (2009) 17–29.
[3] F. Castro, À. Nebot, F. Mugica, On the extraction of decision support rules from fuzzy predictive models, Appl. Soft Comput. J. 11 (4) (2011) 3463–3475.
[4] N. Collins, Experiments with a new customisable interactive evolution framework, Organised Sound 7 (3) (2002) 267.
[5] N. Collins, Interactive evolution of breakbeat cut sequences, Proceedings of Cybersonics, 2002.
[6] N. Collins, A. McLean, J. Rohrhuber, A. Ward, Live coding in laptop performance, Organised Sound 8 (3) (2003) 321.
[7] D. Cope, Algorithmic Music Composition, Springer Netherlands, Dordrecht, pp. 405–416.
[8] P. Dahlstedt, Creating and exploring huge parameter spaces: interactive evolution as a tool for sound generation., ICMC, 2001.
[9] R. Dawkins, The Blind Watchmaker: Why the Evidence of Evolution Reveals AUniverse Without Design, WW Norton & Company, 1986.
[10] R.T. Dean, The Oxford Handbook of Computer Music, Oxford Handbooks, Oxford University Press, 2009.
[11] S. Goldmann, Presets - Digital Shortcuts to Sound, The Bookworm, an imprint of The Tapeworm, 2015.
[12] H. Helmholtz, On the Sensations of Tone, Dover Publications, INC., New York, 1954.
[13] A. Hunt, R. Kirk, Mapping strategies for musical performance, in: Trends Gestural Control Music, 21, 2000, pp. 231–258.
[14] L. Kuncheva, Fuzzy Classifier Design, Physica-Verlag Heidelberg, 2000.
[15] R. Likert, A technique for the measurement of attitudes, The Science Press, New York, 1932. Ph. D. Columbia University, Archives of psychology, no. 140.
[16] T. Magnusson, Herding cats: observing live coding in the wild, Comput. Music J. 38 (1) (2015) 8–16.
[17] J. McCartney, SuperCollider: a new real time synthesis language, in: Proceedings of International Computer Music Conference, International Computer Music Association, 1996, pp. 257–258.
[18] J. McCormack, A. Eldridge, A. Dorin, P. McIlwain, 2011. Generative Algorithms for Making Music: Emergence, Evolution and Ecosystems, Oxford University Press,New York, pp. 354–379.
[19] E.R. Miranda, Granular synthesis of sounds by means of a cellular automaton, Leonardo 28 (4) (1995) 297–300.
[20] G. Nierhaus, Algorithmic Composition: Paradigms of Automated Music Generation, Springer Science & Business Media, 2009.
[21] I. Paz, Parametric perceptual exploration, 2017, (https://github.com/ivan-paz/parametric-perceptual-exploration).
[22] I. Paz, M. Nebot, E. Romero, F. MÃºgica, A. Vellido, A methodological approach for algorithmic composition systems' parameter spaces aesthetic exploration., in: IEEE Congress on Evolutionary Computation., in: CEC '2017, 2017, pp. 1317–1323.
[23] M. Pearce, D. Meredith, G. Wiggins, Motivations and methodologies for automation of the compositional process, Music. Sci. 6 (2002) 119–147.
[24] C. Roads, Sound composition with pulsars, J. Audio Eng. Soc 49 (2001) 134–147.
[25] J. Rohrhuber, A. de Campo, Improvising formalisation: conversational programming and live coding, New Comput. Paradig. Comput. Music. Sampzon, Fr. Delatour (2009).
[26] D. Stowell, A. Robertson, N. Bryan-Kinns, M. Plumbley, Evaluation of live human-computer music-making: quantitative and qualitative approaches, Int. J. Hum. Comput. Stud. 67 (2009) 960–975.
[27] M. Wanderley, N. Orio, Evaluation of input devices for musical expression: borrowing tools from HCI, Comput. Music J. 26 (2002) 62–76.
[28] G. Wang, P. Cook, On-the-fly programming: using code as an expressive musical instrument, in: Proceedings of 2004 Conference on New Interfaces MusicalExpression, in: NIME '04, 2004, pp. 138–143.
[29] S. Wilson, D. Cottle, N. Collins, The Supercollider Book, MIT Press, Cambridge, MA, 2011.