MULTI-SCALE ORACLE AND AUTOMATED REPRESENTATION OF FORMAL DIAGRAMS BASED ON THE COGNITIVE ALGORITHM

Joséphine Calandra
IReMus, LaBRI
josephine.calandra@labri.fr

Jean-Marc Chouvel
IReMus
jeanmarc.chouvel@free.fr

Myriam Desainte-Catherine
LaBRI
myriam@labri.fr

ABSTRACT

This article deals with the automatic representation of formal diagrams, which corresponds to a paradigmatic analysis of the musical work which is being listened to. These diagrams represent musical materials as a function of time and are initially obtained from the audio signal, applying a *Cognitive Algorithm*. In this article, we focus on the second step of the algorithm, as such we assume that the first step, analyzing and labeling the audio signal, has been done. Thus, we propose to analyze predefined materials given as a string. Then we develop the automatic creation of all the formal diagrams of higher levels that result from it. The structuration of the sequences of materials of the lower level constructs the formal diagrams of higher levels. The structured characters which are gathered then represent a higher-level material. Therefore, we present the Multi-scale Oracle: a data structure that stores and connects the different levels and materials. Thus, a character string given as input of the system produces a superposition of formal diagrams as a function of various structuring parameters. As the hierarchical formal diagrams offer a new representation of music, we suggest the musicologists could use these diagrams for analysis.

1. INTRODUCTION

Our research consists of performing a music analysis as a cognitive process accurately related to this specific music as a phenomenon in time. Therefore, we seek to model the representation of the temporal evolution of a piece of music according to the different cognitive imperatives of people who are listening to music. As it has been demonstrated by François Delalande in [1], cognitive imperatives induce the way people listen to music and therefore structure music. By offering in this method of analysis a different representation, thus a different than commonly used reading, we want to understand how music creations through their audio representation are segmented by the listeners and therefore procure emotions to them (G. Brelet [2], J. Sloboda [3]). Listening systems and representations over time, such as OSSIA Score [4] and Antescofo [5], exist but do not offer a structured representation. On the other hand, the structure estimation of pieces of music by multi-criteria analysis and regularity constraint developed by G. Sargent in his thesis [6] gives hierarchical representations that do not consider the temporal phenomenon.

In previous work, one of the authors [7] developed the *Cognitive Algorithm*: a listening model which aims at modelizing the cognitive processes that are set up while listening to music. Nevertheless, this modelization is a hypothetical one that structures the music listener's audio representation to be as faithful as possible to the *Gestalt Theory*. By *structure*, we mean the action of segmenting in a way that gathers objects together to constitute a unique object of higher level. The output is a representation of formal diagrams at different structuring levels of distinct time scales. Each formal diagram gives a view of the music constituents' temporal evolution, called *objects* that are the elements analyzed as instantiated paradigms. We draw a line between the *object* and the *material* as the latter is the actual paradigm. For instance, the deployment of new materials is highlighted by the discovery front: the line representing the most recent new materials at each instant t. Formal diagrams also highlight behaviors such as pattern reiteration, reversion, or speed. From a given-structural-level diagram, and in parallel with the construction of this diagram, we can create the higher-level one by using segmentation and similarity criteria and, therefore, grouping lower-level materials to create a higher-level material. Moreover, the analysis and then the formal diagram representation is not specific to the note scale but applicable to different temporal scales and different sound dimensions. These are determined by the similarity and segmentation criteria provided as a parameter of the system and the constitution of lower-level diagrams.

Besides, this representation is different from scores or piano-rolls as it changes the ordinate axis and highlights the temporal appearance of the materials and not the pitches, which offers another reading for pieces of music. A form of compression is also obtained as the material memory increases as we listen to music. Then, to get an accurate representation, we do not need to know all the materials when processing the music representation. Therefore, there is a need to determine the similarity and segmentation criteria that will be put in place to run a configurable algorithm as generic as possible to obtain as many representations as possible perceptions of music. In this article, we want to develop an algorithm that can offer representations that a musicologist would not have been able to create from the angle of his own experience and analysis by automating the *Cognitive Algorithm*. Even if this algorithm has already been tested manually by Jean-Marc Chouvel, our goal here is to automate this analysis in order not only to simplify the task of the musicologist but also to remove one's pre-established knowledge to better understand the composition and auditory processes.

In this article, we focus on the second part of the algorithm, assuming that the audio signal is already labeled into characters. Then we will deal with the analysis of character strings representing the music. F. Lerdahl and R. Jackendoff [8] proposed a theory to structure tonal music with different grouping rules that would bring some tools for the elaboration of segmentation rules in our algorithm.

Memory is where the information is stored and the question of access to the stored elements and memory optimization arises: this is why there is a need to augment the *Cognitive Algorithm* with an accurate representation of the memory. For that, we propose to use the *Factor Oracle* as implemented by C. Wang and S. Dubnov [9] and first presented by C. Allauzen, M. Crochemore, and M. Raffinot in [10].

We seek to create a multi-scale computed representation that gathers different superposed reading levels. Moreover, we aim to do so by systemizing the paradigmatic and syntagmatic analysis optimally. However, we are not interested in the semiotic analysis of these results: we leave this work to the musicologists wishing to analyze the composition put in the system.

In Section 2, we explain how the *Multi-scale Oracle* works, and what data structures we chose to organize the memory of the modelized listening subject. In Section 3 we will explicit the different rules implemented to structure hierarchically the various character strings that are input. Section 4 explains the structuring function implementation and the memory hierarchization, and Section 5 presents applications of the *Cognitive Algorithm* to Wolgang Amadeus Mozart's K545 *Rondo* and Claude Debussy's *Hommage à Rameau* reduced as label strings. Then, we will discuss the results in Section 6.

2. MULTI-SCALE ORACLE

A description of the *Multi-scale Oracle*, the combination of the *Cognitive Algorithm* and the *Factor Oracle*, is provided. We also describe the chosen data structures for memory implementation, where the materials are stored.

2.1 Presentation of the Cognitive Algorithm

The *Cognitive Algorithm* (Figure 1) was developed and applied by one of the authors [11] to propose a methodological algorithm for musical analysis. This algorithm is the process that constructs the different layers of formal diagrams associated with the music supplied as input. This is mainly composed of two tests. A first test is called *Paradigmatic Recognition Test*. It is a comparison between the object currently being listened to and the previously listened to objects. The second test is the *Syntagmatic Recognition Test*. This test validates whether the concatenation

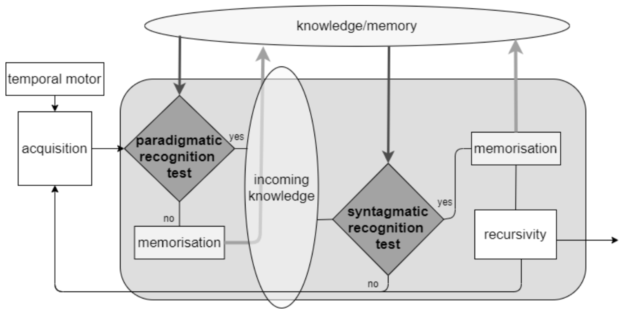


Figure 1: General Scheme of the *Cognitive Algorithm*. For more information, a complete version can be found in [12].

of the objects obtained since the last segmentation constitutes a higher-level object.

The algorithm at a level of structure i is the following. All the objects are processed one by one. At the instant t, the object $obj(i, t)$ is assimilated. The first test is then carried out: if the object which has just been heard is similar to an object previously heard of the same level, then it is recorded in the *formal memory*. The *formal memory* corresponds to the memory where the objects in the time dimension are stored. If this object has never been heard before, it is written in the *formal memory* and also in the *material memory*: the memory that stores every new material which appears over time. Then, the current object $obj(i, t)$ is concatenated in the *incoming knowledge* with previously heard objects of the same level. The assumption is made that a higher level object is being created and the next objects at this level can be guessed according to what has already been processed before. Then comes the second test: if there is at this moment a structuring criterion, then the test is validated and this hypothetical object becomes a higher-level object. The structuring criteria will be clarified in part 3.2. Then, we iterate the same algorithm with this new higher-level object $obj(i + 1, t - k)$, where k is the number of lower-level concatenated objects multiplied by their duration, as input. If, conversely, no segmentation criterion is detected, the test is invalidated and the algorithm is reiterated with the next object of the same level.

The difficulty of the computer implementation is that the functions described in this algorithm are general. There is a need to make them mathematically explicit while allowing the system to remain as inclusive as possible because we want to analyze any type of music.

In a previous article [13], we focused on a single level of construction, the first one which is the sampling window scale. We also focused on the first part of the algorithm: the *Paradigmatic Recognition Test* at the signal level. In this article, we leave the signal analysis aside and focus on analyzing from a higher level where materials are already represented by symbols. Therefore the first test corresponds to a character string comparison and the second test corresponds to the structure by segmentation criteria explained in part 3.2.

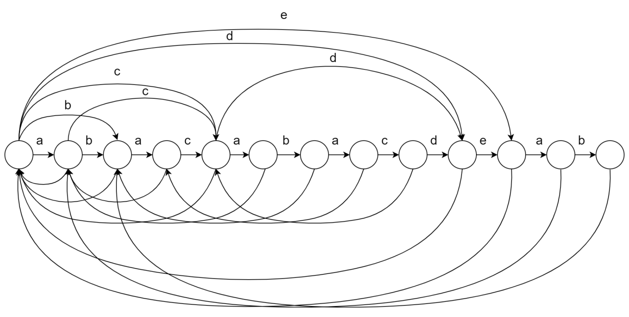


Figure 2: *Factor Oracle* based on the string *"abacabacdeab"*.

2.2 Organization of the memory

The audio is represented by a character string where each character corresponds to a given material. A material can be a declination of any specific musical criterion such as a specific pitch, tone, or even rhythm at different hierarchical levels such as notes scale, grouping notes scale, phrases, or music parts. Therefore the input object for each level of the system is at time t a single character. The memory organization is essential for the system performances because the algorithm performs a constant back and forth, especially with the previously heard objects of the different levels.

2.2.1 Factor Oracle

Objects acquired in the system are represented in memory by an oracle as designed by C. Wang and Shlomo Dubnov [9]. The *Factor Oracle* is a data structure of the form $Q = q_1q_2...q_t...q_T$ with T states. This oracle enables, thanks to *links*, to quickly find the longest character string previously heard in the system, which is similar to the currently heard character string. There are two types of *links*: *suffix links* and *forward links*. *Suffix links* are created from state $t + k$ to state t with $k > 0$ when the element corresponding to the link q_{t+k} going from state $t + k - 1$ to state $t + k$ and the element corresponding to the link q_t going from state $t - 1$ to t are equals. *Forward links* are of two types: the *internal forward links* correspond to the *links* going from the previous state $t - 1$ to the state t and correspond to the temporal unfolding of the musical work. *External forward links* are created from state t to state $t+k$ with $k > 0$ when the most recent *internal forward link* q_{t+k} is preceded by a q_{t+k-1} *link* such that $q_{t+k-1} = q_t$ and q_{t+k} have never followed q_t before. Each *forward link* is named after the index corresponding to the listened material at that moment, the *suffix links* are not labeled and the states correspond to each temporal moment t of the input. We find an example of a *Factor Oracle* created from the string *"abacabacdeab"* in Figure 2.

For the implementation of the *Factor Oracle*, we use the code developed by C. Wang and S. Dubnov [14].

2.2.2 Relation between the Oracle and the Cognitive Algorithm: the Multi-Scale Oracle

The benefit of using such a memory organization is that there is no need to go through the entire string to find previous similar elements. Searching and comparing elements in the entire string has a $nlog(n)$ complexity. With the *Factor Oracle*, accessing the last similar item is constant once the new state is created. Nevertheless, the creation of the oracle data structure still requires computation time.

Also, the oracle models the *incoming knowledge memory*: indeed, thanks to the *forward links* of the currently heard object suffix, we can make one or more hypotheses about what the oracle expects to follow this object. These hypotheses will be validated or not by *Syntagmatic Recognition Test*. Whether or not these hypotheses are validated provides information on the auditory cognitive behavior by consolidating the music structure or, conversely, by creating a surprise effect.

Nevertheless, the oracle does not compute any information about segmentation, so we need to create rules to justify segmentation and we need to create data structures to store the materials. The *incoming knowledge memory* and the structuring rules need to access the higher-level objects, such as the oracle anticipates the incoming objects at the actual level according to the existing ones in higher levels that contain these actual-level objects. For that, we create different Factor Oracles at each necessary structure level, and we complete those with other data structures that store the information connecting every level. This is what we call the *Multi-Scale Oracle*.

2.2.3 Multi-Scale Oracle data structures description

Besides the oracle, various data structures are added to memorize the materials and to make links between the different hierarchical levels of the formal music analysis. The *Multi-Scale Oracle* is the concatenation of the *Factor Oracle* and the following structures, instantiated as many times as there are hierarchical levels of structure.

The *concatenated object* corresponds to the word currently created in the *incoming knowledge memory* for the higher level. This character string corresponds to the concatenation of all the characters read since the last structure in this level. Each time the string is structured, this character string is reinitialized to the empty word, then each new character corresponding to the material heard is concatenated until it is structured again.

The *links table* associates each current state with the index of the corresponding state of the higher-level oracle. This table is updated at each structure. When structuring, all the characters contained in the *concatenated object*, which correspond to different states at the level of the current oracle, correspond to a single character at the higher level. Therefore as many indices as there are characters contained in the *concatenated object* are added in the *links table*. These indices have the same value, which is the maximum index of the *links table* incremented by 1. This value corresponds to the index of the higher-level state associated with the new structured object.

The *history next table* corresponds to all the higher-level materials obtained until instant *t*. This is an array of pairs (i_label, i_str) where i_label corresponds to the higher-level label and i_str is the corresponding current level structured string. This data structure is optimized because it is updated in constant time and browsed in maximum linear

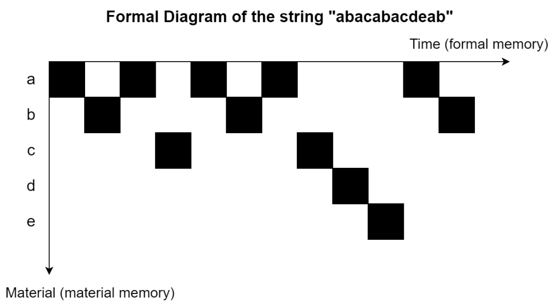

Formal Diagram of the string "abacabacdeab"
Time (formal memory)
a
b
c
d
e
Material (material memory)

Figure 3: Formal diagram of the string *"abacabacdeab"*.

time meanwhile it contains at most as many elements as the higher level.

The *formal diagram* corresponds to the formal representation of the current state of the oracle at the current hierarchical level. This diagram is a matrix of size $n * t$ where n is the number of materials and t is the current time. For each index $M_{i,j}$, with $0 \leq i < n$ and $0 \leq j < t$ we have $M_{i,j} = 1$ if at instant j the material i is parsed, $M_{i,j} = 0$ otherwise. An example of a formal diagram created from the string *"abacabacdeab"* is shown in Figure 3.

3. CREATION AND IMPLEMENTATION OF THE TWO PARADIGMATIC AND SYNTAGMATIC RECOGNITION TESTS

In this section we present how the different rules of the *Paradigmatic* and *Syntagmatic Recognition Tests* were determined and implemented.

3.1 Paradigmatic Recognition Test

The first test compares the current object as strings with the previously heard objects. At the moment, this is a module verifying the strict equality between two strings. Of course, we intend to improve this part in the future because strict equality between two objects considered to be similar in a musical piece is rare. Two patterns considered as similar can, in reality, have a variation of note, rhythm, tempo, nuance, or even transposition, which can be transcribed as a different character at a lower level and thus also impact the recognition of the pattern at higher levels.

For example, the patterns *C-C-G-G-E-E-C-C* might be considered similar to *C-G-E-C* for the musicologist while the strict comparison of the associated strings *aabbccaa* and *abca* (with *a* associated to *C*, *b* to *G*, and *c* to *E*) gives a negative result for sure. The same problem will arise with time expansion or reduction of the same pattern, or *nota cambiata* that might appear in some patterns. The question of computing similarity will then arise in later studies.

3.2 Syntagmatic Recognition Test

We set up different structuring rules for the second test. Each rule was developed according to the constraints imposed by the previously implemented ones. We searched the most basics rules considering the more complex F. Lerdhal and R. Jackendoff generative theory [8], based on the Gestalt theory. When the second test is raised, these rules are applied or not according to the current context: if the rule is applied, the test is validated, otherwise, it is not.

We suggest five rules. Nevertheless, more advanced rules can be developed in the future.

3.2.1 Rule 1

Rule 1: when an already known object at level n is read again, the level n is structured just before this object. The structured *concatenated object* constitutes a higher-level object.

Here is an example of Rule 1 on the string *"abacabacdeabfgabachijklmhinopqabacrsrsttu"*. ***Str. 0-1*** stands for the structure of level 0 giving a corresponding object at level 1. Level 1 contains the labeling of such structure with strict string equality. Opening parenthesis represents the beginning of a new object at a higher level while a closing parenthesis represents the structuring operation.
Level 0 : abacabacdeabfgabachijklmhinopqabacrsrsttu
Str. 0-1 : (ab)(ac)(a)(b)(a)(cde)(a)(bfg)(a)(b)(a)(chijklm)(h)(inopq)(a)(b)(a)(crs)(r)(st)(tu)
Level 1 : ABCDCECFCDCGHICDCJKLM
Str. 1-2 : (ABCD)(CE)(CF)(C)(D)(CGHI)(C)(D)(CJKLM)
Level 2 : ABCDEFDEG
Str. 2-3 : (ABCDEF)(D)(EG)
Level 3 : ABC
Str. 3-4 : (ABC)
Level 4 : A
Correspondance between level 3 and level 0 gives A= (abacabacdeabfgabachijklmhinopq), B= (a) and C= (bacrsrsttu)

However, we see that this rule alone can give disproportionately sized objects at high levels. It is not consistent in a perspective where one would want objects of the same level to have a similar duration (Grouping preference rules 5 in [8]).

Besides, this structuring system also raises concerns because this rule essentially destroys the principle of hypothesis. If we have, for example, the sequence *"abacab"*, there should be segmentation before the second and the third *"a"* because an already known *"a"* appears again and there is also segmentation of the second *"b"* for the same reasons. This gives the structure *"(ab)(ac)(a)(b)"*. This structure destroys the hypothesis expecting a *"b"* after the arrival of the second *"a"*. Therefore, there is no possibility that the word *"ab"* will come up again because this is what happened at the first occurrence of *"a"*.

When we speak about structuring while an already known object is back, it is for the initial segmentation, in the absence of additional rules, such as those of the Gestalt theory. Once there is a grouping, the structuring must indeed be done in real-time with all the activated levels considering the higher-level object and not layer by layer. Therefore, there is no question of segmenting between *"a"* and *"b"* if there has been a group *"(ab)"* recorded (this can be referred to the Grouping Well Formedness Rule number 4 in [8]).

3.2.2 *Rule 2*

Rule 2 aims at resolving the hypothesis problem: level n is structured only if the object constituted of the *concatenated object* plus the character that has just been read does not exist at the higher level.

Therefore, in the previous example, we do not structure before the arrival of the second b. We then have the structure *"(ab)(ac)(ab)"*.

We must specify then that the segmentation is not done one level after the other but all levels at once. First, the only string we know *"abacabacdeabfgabachijklmhinopq abacrsrsttu"* is read in real-time :
step 1 - **Level 0 :** a
step 2 - **Level 0 :** ab
step 3 - **Level 0 :** aba
There is a *"a"* again, so *"ab"* is structured and a new level is created, containing the symbol *"A"* that corresponds to the label of the lower-level group *"(ab)"*. That gives at step 4:
Level 0 : aba
Str. 0-1 : (ab)a
Level 1 : A
Then the first string is read again until the next structuration. We know *"(ab)"* corresponds to a material of level 1 so we will not structure in the middle of this material at level 0. If there is a structuring criterion at level 1 (for example if A is read again), a level 2 is created and computed in parallel.
Step 12 gives the results:
Level 0 : abacaba
Str. 0-1 : (ab)(ac)(ab)a
Level 1 : ABA
Str. 1-2 : (AB)A
Level 2 : A

The string *"abacabacdeabfgabachijklmhinopqabacrsrsttu"* structured with the rules 1 and 2 gives the results :
Level 0 : abacabacdeabfgabachijklmhinopqabacrsrsttu
Str. 0-1 : (ab)(ac)(ab)(acde)(abfg)(ab)(achijklm)(hinopq)(ab)(acrs)(rst)(tu).
Level 1 : ABACDAEFAGHI
Str. 1-2 : (AB)(ACD)(AEF)(AGHI)
Level 2 : ABCD
Str. 2-3 : (ABCD)
Level 3 : A

Nevertheless, in the string *"abacabacde"*, the return of the pattern *"ac"* represented by *"B"* at level one is hidden in a larger pattern *"acde"* because the new object *"d"* does not allow to structure after the second occurrence of *"ac"*: the structure is then *"(ab)(ac)(ab) (acde)"*. Then, we propose Rule 3

3.2.3 *Rule 3*

As soon as the *concatenated object* at level n is an object already seen at the upper level, the string is structured (Grouping Well Formedness Rule number 5 in [8]).

This rule complements Rule 4, which, however, requires additional computations.

3.2.4 *Rule 4*

There is a structuring operation when the *concatenated object* is a string that has already been heard even included in a larger higher-level object. It is then also necessary to structure the element when it was seen for the first time and to modify the objects of higher levels accordingly.

The string *"abacabacdeabfgabachijklmhinopqabacrsrsttu"* structured with the four rules gives the results :
Level 0 : abacabacdeabfgabachijklmhinopqabacrsrsttu
Str. 0-1 : (ab)(ac)(ab)(ac)(de)(ab)(fg)(ab)(ac)(hi)(jklm)(hi)(nopq)(ab)(ac)(rs)(rs)(t)(t)(u).
Level 1 : ABABCADABEFEGABHHIIJ
Str. 1-2 : (AB)(AB)(C)(AD)(AB)(EF)(EG)(AB)(H)(H)(I)(I)(J)
Level 2 : AABCADEAFFGGH
Str. 2-3 : (A)(A)(BC)(A)(DE)(A)(F)(F)(G)(G)(H)
Level 3 : AABACADDEEF
Str. 3-4 : (A)(A)(B)(A)(C)(A)(D)(D)(E)(E)(F)
Level 4 : AABACADDEEF

The difference between Rule 3 and Rule 4 in this example is the structure of *"(hi)"* with Rule 4 while Rule 3 would have given the two objects *"(hijklm)"* and *"(hinopq)"*. However, immediately repeated objects are immediately structured as higher-level materials with this rule. This induces isolation of these materials which can no longer be included in a larger material. This also implies that all the higher levels will contain exactly these same materials of this given length.

Also, the structuring algorithm ends, not because we can no longer structure, but because there is a loop and the same structures are obtained at the highest levels.

3.2.5 *Rule 5*

To avoid this material isolation, Rule 5 is therefore proposed: an isolated object is not structured (Grouping Preference Rule 1 in [8]).

The string *"abacabacdeabfgabachijklmhinopqabacrsrsttu"* structured with the five rules gives the results :
Level 0 : abacabacdeabfgabachijklmhinopqabacrsrsttu
Str. 0-1 : (ab)(ac)(ab)(ac)(de)(ab)(fg)(ab)(ac)(hi)(jklm)(hi)(nopq)(ab)(ac)(rs)(rs)(ttu).
Level 1 : ABABCADABEFEGABHHI
Str. 1-2 : (AB)(AB)(CAD)(AB)(EF)(EG)(AB)(HHI)
Level 2 : AABACDAE
Str. 2-3 : (AAB)(ACD)(AE)
Level 3 : ABC
Str. 3-4 : (ABC)
Level 4 : A

In this case, structured elements of roughly equivalent length are obtained at each level. However, this rule might not be efficient on the one hand because the isolated elements are integrated into the following grouping whereas it might be more relevant to integrate them into the previous grouping

Algorithm 1: Structuring_function(level n=0, marker m=0)

```
Result: Every hierarchical level structured with the
        associated formal diagrams
if level n does not exist and marker $m == 0$ then
    $oracle_n \leftarrow initialization(oracle_n)$;
    $links_n \leftarrow [0]$;
    $history\_next_n \leftarrow tab(empty)$;
    $formal\_diagram_n \leftarrow matrix(empty)$;
    $concat\_obj_n \leftarrow string(empty)$;
end
while parsing string s do
    $c \leftarrow parsed(s)$;
    $oracle_n \leftarrow add\_state(oracle_n, c)$;
    $formal\_diagram_n(c, t) \leftarrow 1$;
    $display(formal\_diagram_n)$;
    if (RULES are passed and $m == 0$) or
    ($m == 1$ and $len(concat\_obj_n) > 0$) then
        $segmentation(n + 1, m)$;
    end
    $concat\_obj_n \leftarrow append(concat\_obj_n, c)$;
    if $level == 0$ and EOS then
        $m \leftarrow 1$;
    end
    if $m == 1$ then
        $segmentation(n + 1, m)$;
        $concat\_obj_n \leftarrow append(concat\_obj_n, c)$;
    end
end
```

or to keep this object isolated, for example, if it is strongly isolated by a silence.

4. MAIN ALGORITHM AND HIERARCHICAL MEMORY

The *Cognitive Algorithm* is implemented in such a way as to update the *Multi-scale Oracle* and obtain the formal diagrams displayed and updated in real-time.

4.1 Structuring function

The main algorithm consists of calling the *structuring_function(level n, string s, marker m)*. This function is initialized at *level n* (usually set at 0), with the initial *string* s, and marker set at 0. A pseudo-algorithm of the structuring function is provided in the Algorithm 1.

Algorithm 2: segmentation(level n, string s, marker m)

```
$t_{n+1} \leftarrow max(links_n) + 1$;
for i in range(length($concat\_obj_n$)) do
    $links_n \leftarrow append(links_n, t_{n+1})$;
end
$s \leftarrow label(concat\_obj_n)$;
$concat\_obj_n \leftarrow string(empty)$;
$structuring\_function(n + 1, s, m)$;
```

The *structuring_function* is a recursive function divided into three parts. The first part corresponds to the initialization of the various data structures at the current level if they do not exist. The *oracle* at level n $oracle_n$ is initialized with the initializing function provided in the *Variable Markov Oracle module* developed by C. Wang and S. Dubnov [14]. The *links table* $links_n$ is initialized with a node 0 which correspond to the index of the initial state of the oracle. The *history next table* and the *formal diagram* at level n $history\_next_n$ and $formal\_diagram_n$ are initialized with an empty table and an empty matrix and the *concatenated object* at level n $concat\_obj_n$ is initialized with an empty string. Then the function starts a loop which parses the character string input to the function. It adds in the data structures the new object acquired at this level. First, a state corresponding to the character *c* parsed on the string *s* is added to $oracle_n$ with the adequate function provided in the *Variable Markove Oracle module* [14]. Then the $formal\_diagram_n$ is updated, meaning that the material corresponding to the character *c* is added to the formal diagram if it does not exist yet, and the value of the matrix at material *c* and time *t* is set to 1. Then the updated $formal\_diagram_n$ is displayed.

In a second part, there is a structuring operation if the previous rules are validated for the character *c* at time *t* (Algorithm 2). The *links table* $links_n$ is updated with the corresponding index of the higher-level node. There is then a comparison of the $concat\_obj_n$ with the already existing top-level objects contained in $history\_next_n$ and *s* become the label at the next level corresponding to the $concat\_obj_n$.

If the object does not exist, $history\_next_n$ is updated with the couple $(s, concat\_obj_n)$ with s as a new label. The *concatenated object* is reset to the empty character string and $structuring\_function(n + 1, s, m)$ is called at the next level with the new labeled object as the input string. At the upper levels, the character string corresponds to a single character *s*, and the function exit and goes back down to the lower level when there is no structuring. If the function is called at the first level of structure, the character string is read as long as there is no structuring. Finally, $concat\_obj_n$ is updated by concatenating the parsed character *c*.

A question is to know when to end the algorithm at higher levels without creating an infinite number of oracles. For this, we use a *marker*. This leads us to the third part: when the last character of the initial character string is read, the marker changes from zero to one and remains at one for each entry in the function at higher levels. Structuring at each level is then enforced. The function $segmentation$ is called again for the last characters of each level so that a new state is not created for this level because it corresponds to the last structuration. Moreover, if the upper level does not exist, it is not created and the algorithm goes directly to the lower level. As all the needed information is provided to the system at this point, the reading and structuring of the remaining characters in the intermediate levels are completed a few steps later.

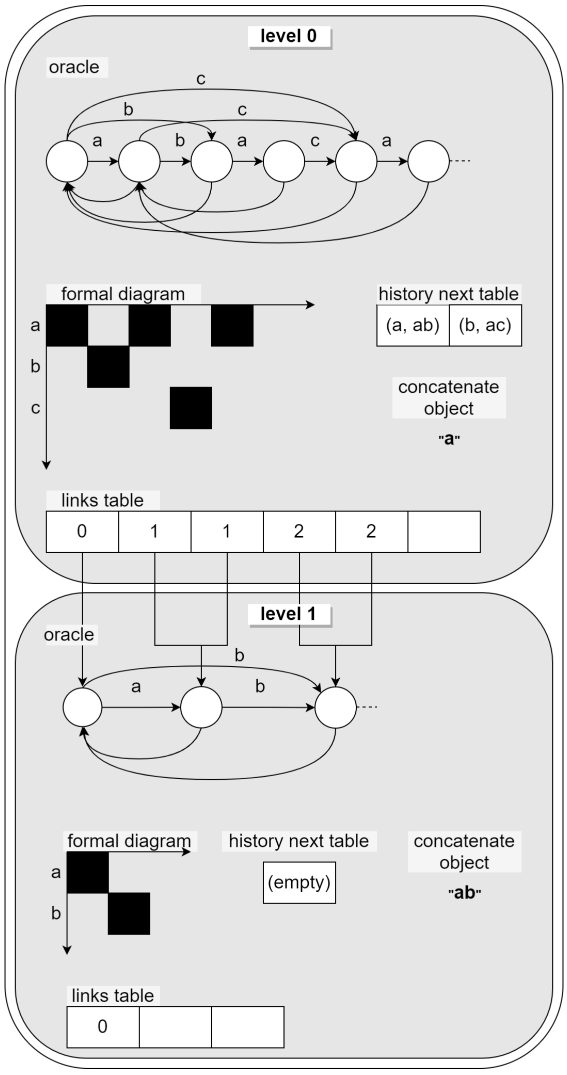


Figure 4: Representation of the *Multi-scale Oracle*.

4.2 Hierarchization of the memory

The organization of the *Multi-scale Oracle* and the back and forth in the memory between hierarchical levels is obtained as follows: the *Multi-scale Oracle* lists for every level *i* a substructure gathering the five data structures described above: a *factor oracle*, a *formal diagram*, a *history next table*, a *concatenated object* and a *links table*. These data structures are updated at any time *t* when new characters are read. Information from higher and lower-level oracles can be accessed through the *links tables*. Relations between the lower-level and higher-level materials are recensed in each *history next* structure. A representation of the *Multi-scale Oracle* is presented in Figure 4.

Each time the lower level is structured, the higher level is created if it does not exist yet and a new state at this higher level is created. The same process is repeated if there is a structuring operation following the creation of this new state. If there is no structuring, the algorithm returns to the lower level where a new state is added, which will lead or not to new structuring operations. If the algorithm is at the lowest level, then it keeps on acquiring new elements of the musical piece, meaning characters from the string.

Figure 5 illustrates schematically the steps of updates and structuration in the *Multi-scale Oracle* according to the five rules with the string *"abacabacdeab"*. First, there is the creation of an *initial state* for the data structures of level 0, meaning the data structures are created and initialized. This is the step (0), the *step of the creation of a new level*. Then, the first character *"a"* is read. A state corresponding to character *"a"* is created in the oracle of level 0 and the *formal diagram* is updated with the material corresponding to *"a"* : this is the step (1), the *step of creation of a state*. There are no structure criteria so anything else happens and *"a"* is appended to the *concatenated object* which was until then the empty word. The same thing is done with the letter *"b"* (2). The third letter *"a"* is read, so a state is added to the according oracle and the *formal diagram* is updated (3). Then, as this letter has already been seen before, there is *structuration* of the *concatenated object* which is *"ab"*. This is the step (4), the *step of structuration*. The next level does not exist yet, so there is the *creation of a new state at the next level*: adequated data structures are created and initialized (5). A new state corresponding to lower-level *concatenated object "ab"*, labelled *"a"*, is created (6) and there are no structure criteria so the loop for level 1 of structure ends, and another character *"c"* is parsed at level 0. The algorithm goes on this way until the end of the string of level 0. The *concatenated object* is structured at states (29), (31), and (33) because the marker is set at 1 since the end of step (28).

5. RESULTS

Examples on W.A. Mozart's *Rondo K.545* and C. Debussy's *Hommage à Rameau* associated strings are provided to illustrate the *Multi-Scale Oracle* analysis based on strings.

5.1 W.A. Mozart's *Rondo K.545*

We take as an example of study the character string *"abacabacdeabfgabachijklmhinopqabacrsrsttu"* which is a reduction of the *Rondo K.545* by W.A. Mozart. We find in Figure 6 the segmentation from the score at the scale of the musical phrase that leads to such a reduction. This reduction was operated manually by the authors and allows us to start on a basis for hierarchical structures automatically computed with the *Cognitive Algorithm*. We did this first analysis from the score to simplify our problem. In the future, the first step will not be made by a musicologist but by a configurable automated audio analyzer giving labels such as what we propose in [13].

We see that the musical phrases labeled with the same letters are simplifications of different variations. What we consider as strict equality during the automatic analysis is then not a real one. The impact of this simplification will be studied when the connection between the signal analysis and the multi-scaled analysis from a character string will be implemented.

As studied in part 4, different results are obtained depending on the rules implemented to structure the piece of music. When we structure according to the five rules mentioned, we find the formal diagrams in Figure 7.

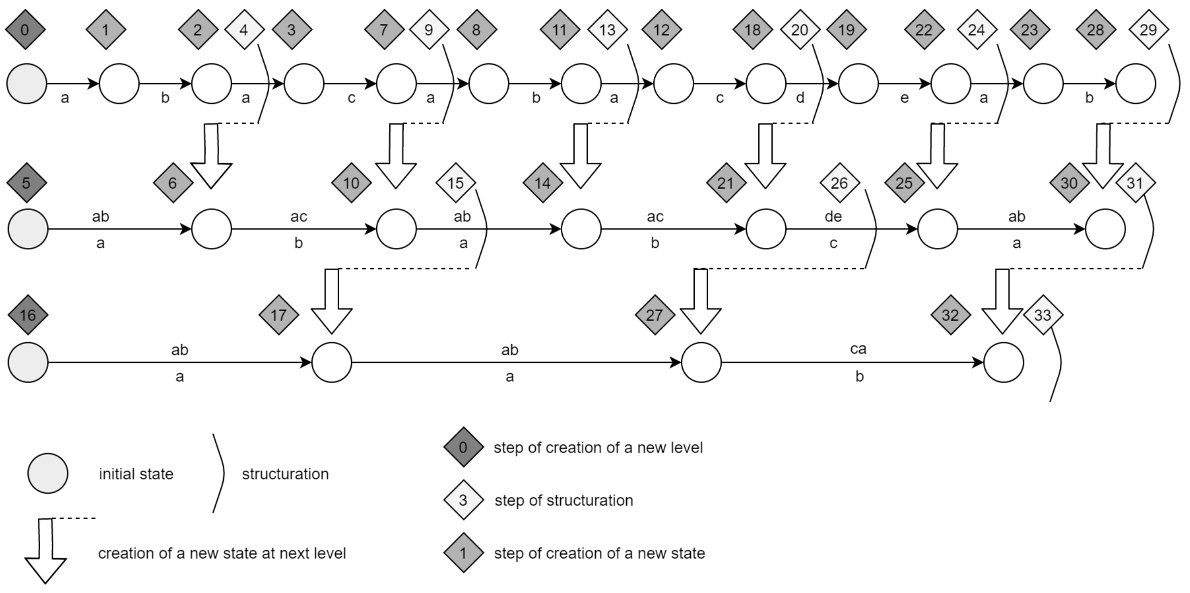


Figure 5: Hierarchization of the memory.

The analysis of these diagrams from a compositional point of view is left to musicologists. However, we can make a few observations about the obtained results. We can compare these diagrams with those obtained when we remove one or more of the stated rules.

If we remove Rule 4 and Rule 5 (see Figure 8) , we end up in the configuration where the diagrams created from a certain level are identical (see part 3.2). So we have as many new diagrams produced as there are oracles created until the end-of-string marker is activated. The termination of the algorithm is therefore ensured, but we have a finite number of identical diagrams and the same number of associated windows for each process that is opened. For example in Figure 4, there are four hierarchical levels, so there are four open windows on the computer's screen that make the results harder to read.

5.2 C. Debussy's *Hommage à Rameau*

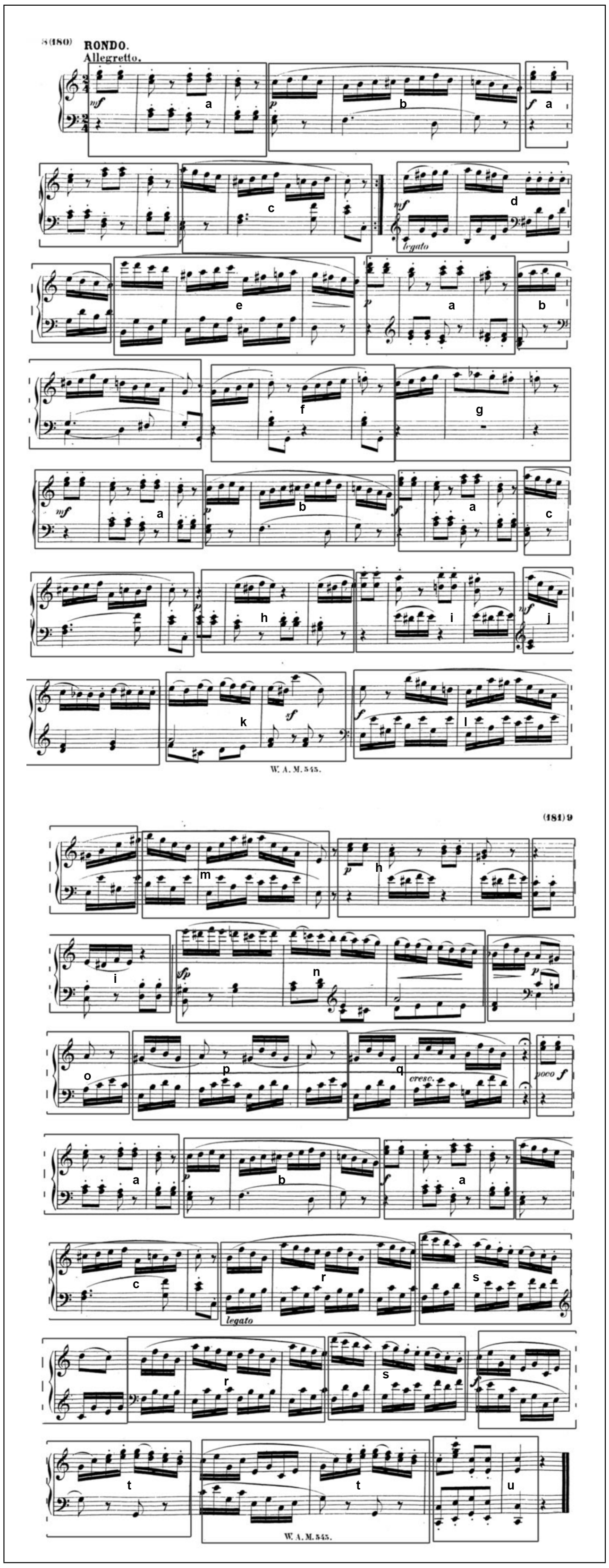
We can run the algorithm with any other string as an entry. From an analysis of Claude Debussy's piece of music *Hommage à Rameau* obtained in the same way than the previous example, the associated labels obtained are *(1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1, 2, 3, 4, 9, 20, 21, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 28, 29, 32, 33, 34, 35, 1, 2, 3, 4, 5, 6, 36, 37, 9, 38, 39, 9, 39, 40, 1, 39, 40, 41, 42)*. As there are more than 26 labels, we use numbers instead of letters for easier readability. The associated hierarchical diagrams obtained with the rules 1, 2, 3, 4, and 5 are represented in Figure 9. The *Paradigmatic Recognition Test* computed with strict equality of the strings gives some rigidity to the formal diagrams which might be more accurate with a more flexible similarity test.

6. DISCUSSIONS

We wonder whether the different rules proposed in part 3.2 are necessary and relevant: the study of W.A. Mozart's *Rondo K.545* gives rather efficient results because it has a relatively conventional form, even if we can already see some ambiguities with structures that play between binary and ternary form. However, some pieces of music can lead to more complex and ambiguous structures. For example, in some configurations, the combined rules 4 and 5 induce a significant restructuration of the character string. Therefore there are additional cognitive operations (in the sense of our modelization, meaning higher processing complexity) with the combination of some rules. Indeed, the extract "abacabachijklmabach" is structured in *"(ab)(ac)(ab)(ac) (hijklm)(ab)(ac)(h"* or *"ABABCAB"* which itself is structured *"(AB)(AB)(CAB"*. The parenthesis is not closed at the end of the strings because the analysis is in progress: the last characters correspond to the *concatenated object* and structuring criteria are required for the characters to be structured. When the *"inopq"* objects are concatenated to the previous string there are changes in the structure. In fact, the *"abacabachijklmabachinopq"* character string becomes *"(ab)(ac)(ab)(ac)(hi)(jklm)(ab)(ac)(hi)(nopq"* that is labeled at higher level *"ABABCDABC"* which itself is structured *"(AB)(AB)(CD) (AB)(C"*. Rule 5, which prevented at first glance from structuring before the third *"A"* in level two of the first example, was finally not used because a new character *"D"* appears with the use of Rule 4 in the second example. Moreover, contrary to the example of W.A. Mozart's *Rondo K.545*, the use of Rule 4 sometimes might not involve only the direct higher level but also other levels. In the example presented before, the combination of several rules induces a deeper modification of several higher levels.

However, these additional computations might not be a problem as maybe the cognitive processes themselves consist of recomputing at any time and restructuring the incoming information.

Another question is to know whether these rules are sufficient. For example, we do not take the time signature into account while it can be significant. The character string



8(180)
RONDO.
Allegretto.
a b a c d e a b f g a b a c h i j k l
W. A. M. 545.
(181)9
m h i n o p q r s a b a c r s t t u
W. A. M. 545.

Figure 6: Structuration of the W.A. Mozart's *Rondo K.545* from the sheet.

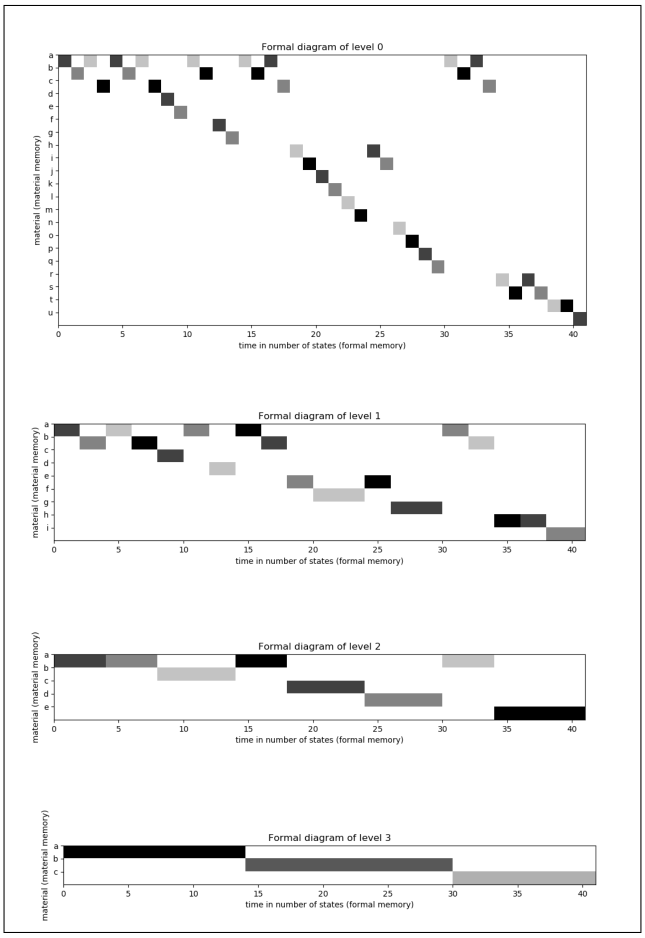


Figure 7: Formal diagrams of W.A. Mozart's *Rondo K.545* at hierarchy levels 0, 1, 2, and 3 with the five structuring rules. The objects have different colors to differentiate two different objects of the same material that are one after the other.

"abacac" may be structured *"(ab)(ac)(ac)"* (binary structuration obtained with the rules we propose in this article) or *"(aba)(cac)"* (ternary slicing). There may even be beat or time signature changes within the same song.

Finally, it is necessary to measure to what extent the implemented rules remain neutral and are not inferred from the knowledge of the musicologist. Adding a time signature, for example, could orient the analysis of a piece where the composer was trying to override its existence in his work.

The computation of similarity between two objects also has to be deepened. Indeed, we want to set up a similarity threshold with a computing distance between two character strings rather than strict equality.

Also, we would like to spell out different functions, such as changes in tempo, nuances, transposition, inversion, and their associated coefficients, to clarify the interconnections between two elements considered to be of the same class. Therefore all the information necessary for the reconstruction of the initial music from a high hierarchical level would be obtained. Moreover, we could represent the presential rate of a previously heard object. For example, if the currently heard object is 80% similar to a previous one, the

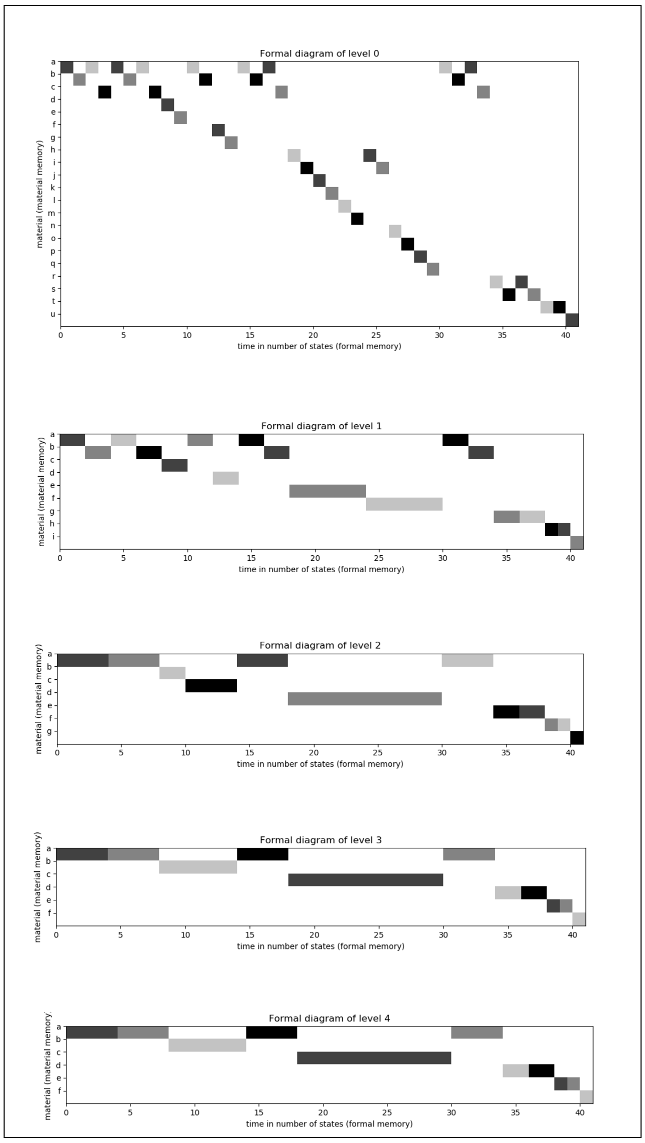


Figure 8: Formal diagrams of W.A. Mozart's *Rondo K.545* at hierarchy levels 0, 1, 2, 3, and 4 with the structuring rules 1, 2, and 3. Six other levels (levels 5, 6, 7, 8, 9, and 10) are created but not shown here as they are identical to levels 3 and 4.

adequate material will be represented accordingly.

Currently, a display is presented in such a way as a window corresponds to a diagram. However, this display is not optimal because there can be a wide range of hierarchical levels meaning a significant number of windows. Furthermore, this display does not align the origins of the different diagrams, while this would visually highlight even more the relations between the different hierarchical levels. In the future, we would like to represent all of these diagrams on a single three-dimensional graph, where the first two axes would be the time axis and the material axis, and the third axis would correspond to the time scale of the structures. It would enable a continuous representation of the structures on different levels. However, we would have to modify our multi-scale oracle, such as it would be less linear for each structuring level. Instead of having one

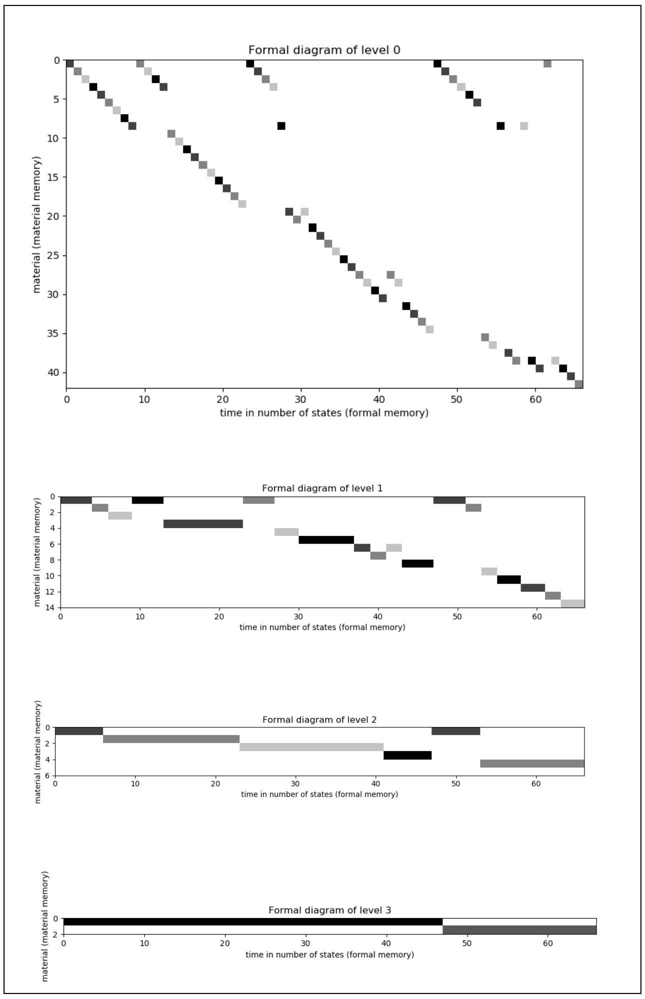


Figure 9: Formal diagrams obtained with the rules 1, 2, 3, 4, and 5 of C. Debussy's *Hommage à Rameau* at hierarchy levels 0, 1, 2 and 3.

augmented *Factor Oracle* at each level, we would have an oracle with multi-scale connexions.

Moreover, at the first level of structuring (from the signal) a representation in gray level for the dynamics of the different objects is implemented, but we still have to integrate it for the different hierarchical levels as there is so far no information on the dynamics with only character strings as input.

7. CONCLUSIONS

In this article, we propose an automatic implementation of a new musical representation. This representation is computed based on the temporal evolution of the piece on different time scales. Then, we suggest an algorithm allowing from a character string, where each character corresponds to a defined musical material, to produce formal diagrams, meaning representations of musical materials as a function of time. These diagrams are structured on different hierarchical temporal levels. Each level is structured in real-time when listening to music, the auditory process being modeled by the parsing of the initial character chain. This automated production of the representation was implemented by the creation of elementary rules defined in this article. We also leave the possibility to choose the *Syntagmatic Recognition rules* to obtain different possible diagrams. Moreover, this article also describes the memory arrangement and the relations between the different levels of memory with the *Multi-Scale Oracle*. Also, we describe and explain the main loop of the algorithm and the trajectory between the different structural levels.

Next, we need to connect the signal analysis of the first level and the entire hierarchical analysis based on string characters. In the first situation, the similarity between objects is computed based on the sampling widows, and the objects are segmented when there are changes. On the opposite, the similarity between strings is calculated based on string comparison, and these are segmented when they are similar. The whole point will now be to create consistency between the two different analyses.

8. REFERENCES

[1] F. Delalande, *La musique au-delà des notes*. Presse Universitaires de Rennes, 2019.

[2] G. Brelet, *le temps musical: essai d'une esthétique nouvelle de la musique, 2 Volumes*. Presses Universitaires de France, Paris, 1949.

[3] J. Sloboda, "Music structure and emotional response: Some empirical findings," *Psychology of Music*, vol. 19, pp. 110–120, 1991.

[4] J.-M. Celerier, P. Baltazar, C. Bossut, N. Vuaille, J.-M. Couturier, and al., "Ossia: Towards a unified interface for scoring time and interaction," in *Proc. Int. Conf. on Technologies for Music Notation and Representation - TENOR 2015*, Paris, France, 2015.

[5] A. Cont, "Antescofo: Anticipatory synchronization and control of interactive parameters in computer music." in *International Computer Music Conference (ICMC)*, Belfast, Ireland, 2008, pp. p.33–40.

[6] G. Sargent, "Estimation de la structure de morceaux de musique par analyse multi-critères et contrainte de régularité," *PhD Thesis, Rennes University*, 2013.

[7] J.-M. Chouvel, "Musical form, from a model of hearing to an analytic procedure," *Interface*, vol. 22, pp. 99–117, 1993.

[8] F. Lerdhal and R. Jackendoff, *A generative theory of tonal music*. Cambridge, Mass. : MIT Press, 1983.

[9] C. Wang and S. Dubnov, "The variable markov oracle: Algorithms for human gesture applications," in *IEEE Multimedia, vol. 22, no. 4*, 2015, doi: 10.1109/MMUL.2015.76., pp. 52–67.

[10] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: a new structure for pattern matching; ; theory and practice of informatics." in *Proceedings of SOFSEM'99*, Paris, France, 1999.

[11] J.-M. Chouvel, *Analyse Musicale, sémiologie et cognition des formes temporelles*. L'Harmattan, Paris, 2006.

[12] ——, "Categories and representation in cognitive musical analysis," *Sonus*, vol. 35, pp. 17–35, 2014.

[13] J. Calandra, J.-M. Chouvel, M. Desainte-Catherine, and al., "Génération automatique de diagrammes formels par un algorithme cognitif modulaire: étude préliminaire," in *Journées d'Informatique Musicales (JIM)*, Strasbourg, France, 2020.

[14] C. Wang and S. Dubnov, "Guided music synthesis with variable markov oracle," in *3rd International Workshop on Musical Metacreation, 10th Artificial Intelligence and Interactive Digital Entertainment Conference*, Raleigh, North Carolina, 2014.