Self-Assembly of Musical Representations in MGS

Louis Bigo, Antoine Spicher

Int. Journ. of Unconventional Computing, Vol. 0, pp. 1–18
Reprints available directly from the publisher
Photocopying permitted by license only

Self-Assembly of Musical Representations in MGS

LOUIS BIGO[1,2] AND ANTOINE SPICHER[1]

[1]*LACL, University Paris-Est Créteil, FR*
E-mail: louis.bigo@ircam.fr
[2]*IRCAM UMR 9912 CNRS - UPMC, FR*
E-mail: antoine.spicher@u-pec.fr

Received: January 21, 2013. Accepted: March 11, 2013.

In this paper, we apply morphogenetic processes, namely self-assembly processes, to compute automatically various abstract spaces that can be used to represent and analyze several well-known musical objects (sequence of chords, interval series, etc.). These constructions have been implemented in MGS, an unconventional programming language belonging to the family of spatial computing languages.

Keywords: Self-assembly, abstract cellular complexes, MGS, rewriting, pattern matching, Hamiltonian and Eulerian path, pitch space, chord space, *tonnetz*, all interval series.

1 INTRODUCTION

The algebraic nature of many musical formalizations has been very early assessed: from the equal temperament to canon, algebraic objects have been used to study combinatorial properties and classify musical structures. Recently, a fresh look on these structures has emerged focusing on topological or geometrical representations. For example, one can characterize harmonic paths in orbifolds [5,25] or build topological spaces embedding musical relationships in their neighborhood relationships [14].

Following this line of research, we are interested to harness natural morphogenetic processes for building spatial representations of musical objects. To this aim, our work rely on the use of MGS , a domain specific programming language dedicated to the modeling and the simulation of *dynamical*

systems with a dynamical structure [12]. Numerous applications in system and synthetic biology have been developed in MGS and proved that it is a fruitful unconventional tool for (re-)designing algorithms tackling problems embedded in space or having a spatial extension.

In this paper we propose the use of a self-assembly process for studying two paradigmatic problems in theoretical music. This paper is organized as follows. Section 2 provides a brief introduction to the MGS spatial programming language. Section 2.3 exemplifies the use of the MGS concepts by specifying a self-assembly of polymers. This self-assembly process is then hijacked for musical purpose. In Section 3 a combinatorial space is built up enabling the enumeration and the topological classification of *All-Interval Series* (AIS). Section 4 describes a spatial representation of collections of chords. The paper ends with a conclusion and a discussion about future works.

2 THE MGS PROGRAMMING LANGUAGE

MGS is a *spatial computing* programming language developed to enlighten the importance of space in computations [2,7,21]. MGS concepts are based on well established notions in algebraic topology [17] and relies on the use of rule based functions, called *transformations*, to compute declaratively with spatial data structures, called *topological collections*.

2.1 Topological Collections

In MGS , all data structures are unified under the notion of *topological collection*: an *abstract combinatorial complex* (ACC) labeled with arbitrary values. The ACC acts as a container and the labels as the elements of the data structure.

More precisely, an ACC $K = (\mathsf{C}, \prec, [\cdot])$ is a set C of abstract elements, called *cells* [24], provided with a partial order $\prec$ called the *boundary relation*, and with a *dimension* function $[\cdot] : \mathsf{C} \to \mathbb{N}$ such that for each c and c' in C, $c \prec c' \Rightarrow [c] < [c']$. We write $c \in K$ when a cell c is a cell of C.

A cell of dimension p is called a p-cell: 0-cells are points, 1-cells are edges, 2-cells are polygons (*e.g.*, facets in a mesh), etc. For example, a graph is an ACC composed of 0- and 1-cells. Another example is pictured in Figure 1.

The $(p-1)$-cells c' lower than a p-cell c for the boundary relation $\prec$ are called the *faces* of c and we write $c' \prec c$ or $c > c'$; c is called a *coface* of cells c'. We call *closure* of c in K the sub-complexes $\bar{c} = (\mathsf{C}', \prec \cap\, \mathsf{C}' \times \mathsf{C}', [\cdot])$ where $\mathsf{C}' = \{c' \in \mathsf{C} \mid c' \preceq c\}$.

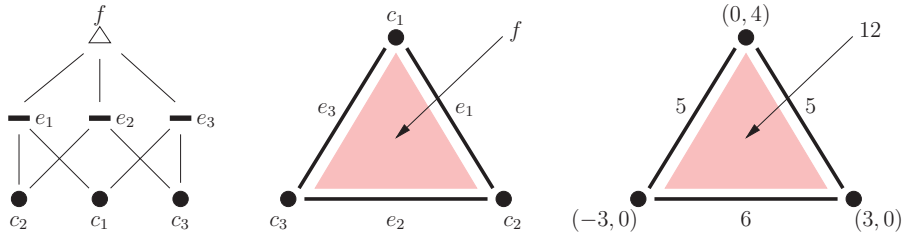


FIGURE 1
On the left, the Hasse diagram of boundary relationship of the ACC given in the middle: it is composed of three 0-cells (c_1, c_2, c_3), of three 1-cells (e_1, e_2, e_3) and of a single 2-cells (f). The three edges are the faces of f, and therefore f is a common coface of e_1, e_2 and e_3. On the right, a topological collection associates data with the cells: positions with vertexes, lengths with edges and area with f.

Two n-cells are (n, p)-*neighbor* if they have a common border of dimension p when $p \leq n$ or if they are in the boundary of a p-cell of higher dimension. A (n, p)-*path* is a sequence of cells such that two consecutive cells are (n, p)-neighbor. For example, the notion of $(0, 1)$-path coincides with the usual notion of path in a graph (a sequence of nodes following a route along some edges of the graph.) We call a (n, p)-*Hamiltonian* path a (n, p)-path visiting each n-cell of an ACC exactly once. Similarly a (n, p)-*Eulerian* path is a (n, p)-path visiting each p-cell of an ACC exactly once.

Finally, a *topological collection* C is a function that associates a value with a cell in an ACC, see Figure 1. Thus the notation $C(c)$ refers to the value of C on cell c. We write $|C|$ for the set of cells for which C is defined. The collection C can be written as a formal sum $\sum_{c \in |C|} v_c \cdot c$ where $v_c \stackrel{\text{df}}{=} C(c)$. With this notation, the underlying ACC is left implicit but can usually be recovered from the context. By convention, when we write a collection C as a sum $C = v_1 \cdot c_1 + \cdots + v_p \cdot c_p$, we insist that all c_i are distinct. Notice that this addition is associative and commutative. This notation is directly used in MGS to build new topological collections on arbitrary ACC of any dimension.

2.2 Transformations

Topological collections are transformed using sets of rules called *transformations* [22]. A rule is a pair `pattern => expression`. When a rule is applied on a topological collection, a sub-collection matching with the `pattern` is replaced by the topological collection computed from `expression`. There exist several ways to control the application of a set of rules on a collection called *rule application strategies*. The present work uses solely the *maximal-parallel* strategy: rules are applied as many times as possible without intersection between matched sub-collections.

A formal specification of topological rewriting is given in [22]. We only sketch here the part of the patterns language necessary for the comprehension. A *pattern variable* specifies a cell to be matched in the topological collection together with some optional guard: the expression `x / x = 3` matches a cell labeled with the value 3. The guard is the predicate after the symbol `/`. The variable `x` can be used in the guard (and elsewhere in the rule) to denote the value of the matched cell or the cell itself, following the context (in case of ambiguity, the variable always denotes the associated value). A pattern is a *composition* of pattern variables. The composition denoted by a simple juxtaposition (e.g., "`x y`") does not constraint the arguments of the composition. Variables can be composed using the (co)face operator: a pattern "`x < y`" (resp. "`x > y`") matches two cells c_x and c_y such that $c_x < c_y$ (resp. $c_x > c_y$). Patterns are *linear*: two distinct pattern variables refer to two distinct cells.

2.3 Illustration: Self-Assembly of Cellular Complexes

MGS is a vehicle used to investigate the notions of topological collections and transformations and to study their adequacy to the simulation of various physical, chemical and biological processes [9–11, 15]. As an example, (local) rewriting rules are particularly adequate to specify self-assembling processes since they mimic closely the incremental and distributed building mechanism of the real phenomenon.

Let illustrate this idea by considering a generic self-assembling process on cellular complexes allowing the building of elaborated spatial structures from a population of basic elements. This process consists in identifying topologically equivalent elements (*i.e.*, cells with the same boundary) in some ACC. This operation is not elementary because the identification must occur at every dimension. A simple way to achieve such computation is to iteratively apply the merge of topological cells that exactly share the same faces until a fixed point is reached. The corresponding topological surgery can be expressed in the MGS syntax as follows:

```
trans Self-Assembly[Pred, Label] = {
   x y / (Pred x y) and (faces x = faces y)
      => let c = new_cell (dim x) (faces x)
                          (union (cofaces x) (cofaces y))
         in  (Label x y) * c
}
```

where the primitive `new_cell p f cf` returns a fresh p-cell with faces f and cofaces cf. The rule specifies that two elements `x` and `y` whose labels check some arbitrary predicate `Pred` (given as a parameter) and having the

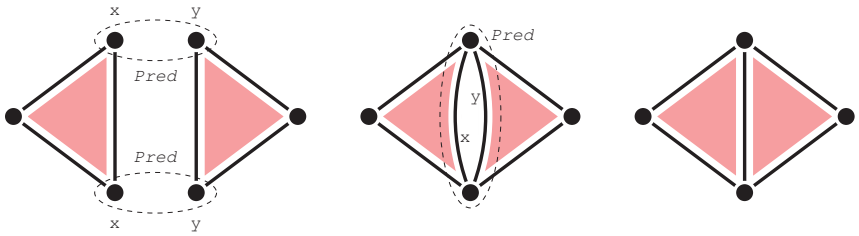


FIGURE 2
Self-assembly of cellular complexes using transformation *Self-Assembly*: at first iteration, two matching pair of nodes are merged; then the resulting edges are identified; finally the fixed point is reached.

same faces in their boundaries, merge into a new cell `c` (that has the union of the cofaces of `x` and `y` as cofaces) labeled by a value computed from some arbitrary function *Label* (given as a parameter). Figure 2 illustrates the process.

Figure 3 illustrates the use of transformation *Self-Assembly* to model a polymerization process. Polymers are long-chained molecules with repeating units called monomers. Monomers react with each other to form polymers in a process called polymerization. Addition polymerization is a particular class of polymerization where monomers combine with polymers in an accretive growth process. Here we consider monomers with two binding sites. A monomer is represented by a rectangular shaped ACC, that is a 2-cell with four edges and four vertices in its boundary. The binding sites correspond to two opposite edges labeled as active. Considering that two active cells can be merged (predicate *Pred*) and that they become inactive after merging (function *Label*), transformation *Self-Assembly* builds up polymers as shown in Figure 3. This toy model does not refer to any natural phenomenon but it could be easily extended to manage with real biological data (*e.g.*, modeling the 3-dimensional structure of some DNA strand.)

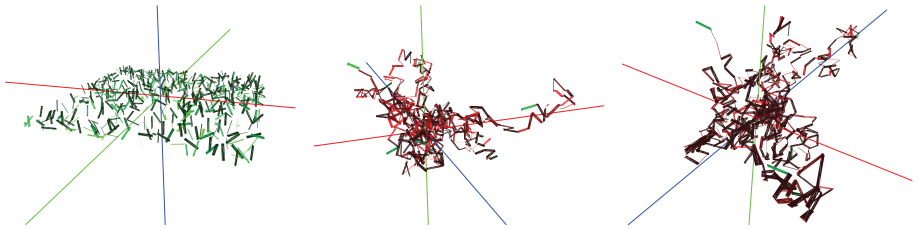


FIGURE 3
Self-assembly of polymers: on the left, the initial population of monomers; on the right, the final polymer seen from two different points of view.

3 SPATIAL INTERPRETATION OF ALL-INTERVAL SERIES

In this section we are interested in the very basics musical notions of *pitch* and *interval*. Some 2D ACC is then elaborated by the self-assembly of spatial representation of the twelve *interval classes*. The combinatorial structure of this space is finally used to enumerate and classify *All-Interval Series* (AIS).

3.1 Backgrounds in Music Theory

In the present work, we are interested in symbolic representation of music. In standard Western music, the usual notation is based on the concept of staff where notes are represented in two dimensions* : vertical height is associated with the *pitch* of the notes (*i.e.*, how high is the associated sound) and horizontal ordering corresponds to time flow (*i.e.*, when notes have to be played.)

In musical analyses, pitches are often considered up to an octave letting us work with only twelve classes called *pitch classes*. For example, the pitch class $C = \{C_0, C_1, C_2, \dots\}$ gathers all the possible Cs. From now on, the term "note" will refer to pitch classes. It is then usual to identify pitch classes to elements of $\mathbb{Z}_{12}$ ($C = 0, C\sharp = 1$, etc.) such that the difference modulo 12 between two pitch classes exactly corresponds to the number of semitones between the corresponding two notes. All the possible differences constitute the *intervals*. For example between $G = 7$ and $D = 2$, the interval is a *perfect fifth* corresponding to $2 - 7 = -5 \equiv (7 \mod 12)$ semitones. In the following, intervals are refered using the usual notation: $P1 = 0$ (*perfect union*), $m2 = 1$ (*minor second* or semitone), $M2 = 2$ (*major second*), $m3 = 3$ (*minor third*), $M3 = 4$ (*major third*), $P4 = 5$ (*perfect forth*), $TT = 6$ (*tritone*), $P5 = 7$ (*perfect fifth*), $m6 = 8$ (*minor sixth*), $M6 = 9$ (*major sixth*), $m7 = 10$ (*minor seventh*), $M7 = 11$ (*major seventh*).

The identification with intergers of $\mathbb{Z}_{12}$ provides the set of intervals with an additive group structure, and the natural action of $(\mathbb{Z}_{12}, +)$ on itself coincides with the *transposition* operation in music. As an example, the action of the perfect forth transpose A into $A + P4 = 9 + 5 \equiv (2 \mod 12) = D$. The action of each interval on the notes is composed of a variable number of orbits. For an interval i it is well known that the number of orbits is $d_i = \gcd(i, 12)$. For instance, there are $d_{m3} = \gcd(3, 12) = 3$ orbits for the minor third:

$$(C - D\sharp - F\sharp - A) \quad (C\sharp - E - G - A\sharp) \quad (D - F - G\sharp - B)$$

These cycles can be uniquely identified by an integer between 0 and $d_i - 1$ corresponding to the least note of the cycle (*e.g.*, C, $C\sharp$ and D for the three cycles above).

* We drop here the consideration of duration.

3.2 Spatial Representation of Interval Classes

In this section, we propose to give a combinatorial construction of the previous formal elements. A usual representation consists of the *all-interval circle* which is a complete graph with twelve nodes, one for each pitch class. Each edge represents the possible transposition from a note to another under the action of an interval. This structure is represented on left of Figure 4. The all-interval circle has two main drawbacks: (1) it does not distinguish intervals and their inverses (*e.g.*, the edge between C and F represents at the same time a perfect fourth a perfect fifth); (2) each interval is represented many times (one for each note).

We propose to elaborate a more complex structure which exhibits these properties by assembling a population of pieces of space called *interval classes*. The interval class I_i is a topological collection representing interval i. The underlying ACC is composed of a unique 2-cell labeled by i. The faces of this cell include twelve 1-cells also labeled by i. Finally twelve 0-cells labeled by the pitch classes constitute the faces of the 1-cells in such a way that two vertices are connected by an edge if their labels are related by the action of i. Top right of Figure 4 illustrates the interval classes associated to perfect fourth and minor third. Note that the boundary of an interval class exibits topologically the orbits of the interval: the number of orbits equals the number of holes $+1$.

The eleven interval classes – perfect unison is not considered – are then glued together by transformation `Self-Assembly` which will identify the note (only 0-cells are concerned) with

```
Pred  x y = (x = y)
Label x y = x
```

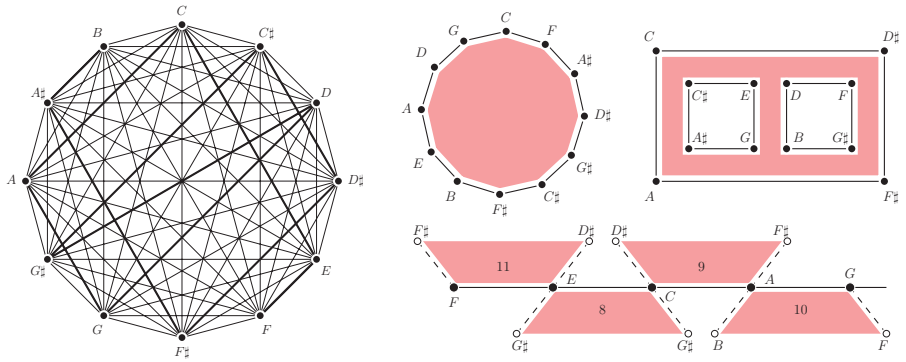


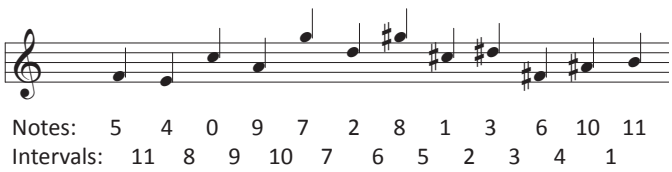
FIGURE 4
Spatial representation of AIS. On the left, complete graph topology with A. Berg's AIS in bold. On the top right, spatial representation of the perfect fourth and minor third interval classes (2-cells are filled in light gray). On the bottom right, spatial representation using interval classes of the five first elements of A. Berg's AIS.

The resulting space is not easily visualizable. Nevertheless, its 1-skeleton (that is the the sub-ACC composed of the n-cells with $n \leq 1$) coincides with the all-interval circle.

3.3 Application to All-Interval Series

The interval class space has interesting properties which allow the study of musical objects from the point of view of combinatorial topology. It is illustrated in next paragraphs with a study of *All-Interval Series* (AIS).

All-Interval Series. An AIS is a twelve-tone series including the eleven different intervals. Such series exhibit many notable properties [16]. One of them is that the first and the last notes are always separated by a tritone interval. One of the most famous use of this particular kind of series is probably on the *Lyric Suite* of Alban Berg:



Other composers like Luigi Nono (e.g., *Il canto sospeso*) or Karlheinz Stockhausen (e.g., *Grüppen*, *Klavierstück IX*) used this material in their compositions.

Beyond the simple enumeration of the 46 272 AIS, composers and music analysts have been interested in finding relevant criterions to classify them. André Riotte proposed a classification considering the harmonic content of the AIS. It consists for example in grouping together AIS containing a subsequence corresponding to the notes of particular scales or chords [19]. Elliot Carter investigated a classification enumerating all AIS containing in sequence the complete set of notes included in the All-Triad Hexachord (this 6-chord is the only one containing the twelve possible triads) [20]. We can also mention an original classification from Franck Jedrzejewski based on knot theory [13].

The enumeration and the classification of AIS is a widely known problem in the computing music community. Several computing approaches, more and more optimized, have been used to enumerate all the AIS. One of the first enumeration was done by André Riotte [19] with the help of a FORTRAN program. This enumeration problem has quickly become a classical problem in Constraint Programming [23] and is now part of the 50 problems of the CSPLib [8]. Some previous works have been based on the enumeration of the All-Interval Chords, which is a similar problem [18].

Spatial Structure of AIS. A naive procedure to collect all the AIS consists in enumerating all the possible permutations of the twelve notes and keeping those exibiting the eleven different intervals. A spatial interpretation of permutation is to look for an Hamiltonian path in the all-interval circle as illustrated in Figure 4 with A. Berg's AIS.

With such a search the 46 272 AIS are lost in the set of 12! candidates. (0, 1)-Hamiltonicity is not a sufficient condition for specifying AIS. An interesting feature of AIS is they are at the same time a permutation of notes and a permutation of intervals. Translated in structural terms, an AIS is a path of the interval class space which is at the same time (0, 1)-Hamiltonian and (0, 2)-Eulerian. It is characterized in a MGS pattern as follows:

```
n0 < i1  < I1  > i1  > n1
   < i2  < I2  > i2  > n2
   ...
   < i11 < I11 > i11 > n11
```

Two consecutive notes $\texttt{n}p'$ and $\texttt{n}p$ with $p' = (p-1)$ have to be (0, 1)-neighbors by some interval $\texttt{i}p$ and (0, 2)-neighbors by some interval class $\texttt{I}p$ such that the interval $\texttt{i}p$ belongs to (*i.e.*, is in the boundary of) interval class $\texttt{I}p$. Linearity of patterns ensures Hamiltonicity. Figure. 4 on bottom right illustrates the instantiation of this pattern.

Topological Classification of AIS. Each AIS visits only one 1-cell in the boundary of each 2-cell and this 1-cell belongs to one of the different orbits associated with the interval class. Let then consider the *cyclic vector* $\mathbf{V} = (v_{m2}, \ldots, v_i, \ldots, v_{M7})$ of an AIS which associates the visited cycle v_i with each interval i. For example, the cyclic vector $\mathbf{V}_B$ of the A. Berg's AIS is

interval class i	$m2$	$M2$	$m3$	$M3$	$P5$	TT	$P5$	$m6$	$M6$	$m7$	$M7$
d_i	1	2	3	4	1	6	1	4	3	2	1
$\mathbf{V}_B$	C	$D\sharp$	C	D	C	D	C	C	C	$C\sharp$	C

AIS can then be classified by gathering in the same class all the AIS sharing the same cyclic vector. This classification has proved to be of great musical interest [4]. For instance, it allows to find AIS including some specific sub-sequence of notes [19,20]. As an example, the interval content of the harmonic C minor scale $C\ D\ E\flat\ F\ G\ A\flat\ B$ invites us to search for some interesting AIS related to that scale with cyclic vector (C, $C\sharp$, D, E, C, F, C, C, $C\sharp$, C, C). For example, this class contains the AIS

$$E \quad D\flat \quad G\flat \quad \boxed{F \quad B \quad D \quad C \quad A\flat \quad E\flat \quad G} \quad A \quad B\flat$$

which exhibits consecutively the seven notes of the scale.

Finally, the proposed classification also allows an easy study of AIS up to the standard algebraic operations [16]: transposition, homothety, retrograde and circular shift. The geometry of the $\prod_i d_i = 3\,456$ possible classes can be folded into a smaller space of only 72 well identified classes.

4 SPATIAL REPRESENTATION OF CHORD COLLECTIONS

Chords play an important role in music theory. In this section, we propose to use the self-assembly process of Section 2.3 to build a combinatorial space representing some collections of chords. We first describe this construction, then we show how various spaces investigated in music theory are recovered.

4.1 Self-Assembly of Chords

A *chord* is a collection of pitches played simultaneously. Depending on the context the collection may have different structures: a set of pitches (*e.g.*, an event of a staff), a sequence of pitches (*e.g.*, in a choral), an ordered set of pitch classes (*e.g.*, chord progressions in Jazz), etc. In the following we will consider chords as set of pitch classes and we call p-chord a chord composed of p notes ($p > 0$).

Chords as Simplicial Complexes. Since a p-chord includes $2^p - 2$ sub-chords, a p-chord can be represented by a topological collection relying on a $(p-1)$-simplex. A *n-simplex* is an ACC of dimension n composed of a unique n-cell c which has exactly $n+1$ faces such that for each face c' the closure of c' is a $(n-1)$-simplex. The 0-cells are labeled by a single note and other p-cells by the corresponding $(p+1)$-chords. Simplices are often represented geometrically as the convex hull of their vertices as shown in Figure 5 for p-simplices with $p \in \{0, 1, 2, 3\}$.

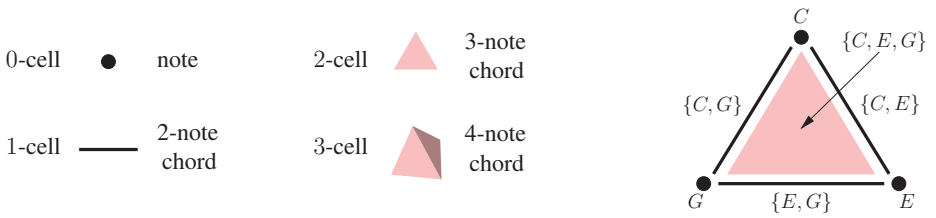


FIGURE 5
A chord represented as a simplex. The complex on the right corresponds to first degree I_C of the C major tonality and all 2-chords and notes included in it.

Self-Assembly of Chords. Transformation `Self-Assembly` may act on a given set of chords by identifying n-cells with common labels:

```
Pred  x y = (x = y)
Label x y = x
```

Although these parameters are similar to the previous example, the identification occurs here at any dimension and will result in a *simplicial complex* (*i.e.*, an ACC made of simplices only).

The following sections give three applications of self-assembly of chords.

4.2 Analysis of a Musical Piece

The self-assembly process is generic enough to represent any set of chords. We propose here to elaborate some space for the study of a piece from its harmonic progression.

Figure 6 shows the simplicial complex resulting from the assembly of chords from the eight first measures of Chopin's Prelude 4 Op.28. The complex exhibits neighborhoods between chords but does not give any information about how these chords are ordered in the prelude. A remarkable fact of this ordering is that only one note differs between two consecutive chords. This property holds on the fourteen chords starting from the second one. Being composed of three-note chords, such a progression corresponds to a (2, 1)-Hamiltonian path in the associated simplicial complex: 1-cells neighborhood between two 2-cells represents the two common notes between the chords. This path is partially presented by black arrows for the five first chords, starting from the second one, in Figure 6. The enumeration of all

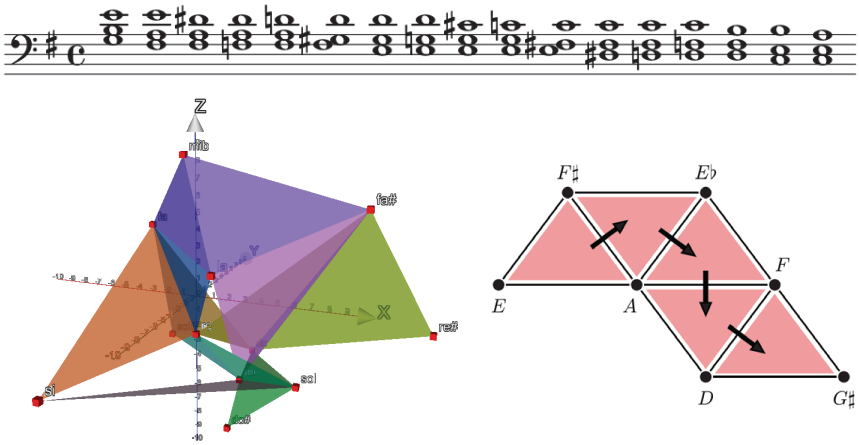


FIGURE 6
Chords of height first measures of Chopin's Prelude 4 Op.28. On the left its simplicial representation. On the right, a path represents the order of chords in a region of the complex.

the possible (2, 1)-Hamiltonian paths in the complex, shows that there exist exactly 120 possible ordering of the chords. But among all these possibilities, the original order used in the Prelude is the one with the smallest distance between chords. Indeed, the interval characterizing the moving note in two consecutive chords is a semitone for all transitions. This example illustrates the topological translation of a well-known compositional strategy called *parsimonious voice leading*. Chord sequences corresponding to other (2, 1)-Hamiltonian in the complex have been generated with MGS and are available in MIDI format at http://www.lacl.fr/ lbigo/aisb13#analysis.

4.3 Tonality Representation

In tonal music, a piece is frequently characterized by the use of successive tonalities. A tonality is defined by a set of notes, each having a particular role. *Triadic chords* (stacked thirds 3-chords) composed by notes of the tonality are called the *degrees* of the tonality. As an example, the seven degrees of the C major tonality are:

$$I_C = \{C, E, G\} \quad II_C = \{D, F, A\} \quad III_C = \{E, G, B\} \quad IV_C = \{F, A, C\}$$
$$V_C = \{G, B, D\} \quad VI_C = \{A, C, E\} \quad VII_C = \{B, D, F\}$$

The self-assembly provides a combinatorial space associated with the tonality. Guérino Mazzola presents in [14] this topological representation of the diatonic tonality which appears to be a Möbius strip (see Figure 7.)

Such a self-assembly process may seem trivial but the elaboration of a space based on chords of higher dimension is difficult by hand. For example tonality may be defined through 4-chords instead of triads (*e.g.* for C major $I_C = \{C, E, G, B\}$, $II_C = \{D, F, A, C\}$, etc.) After computing the Euler characteristic and the orientability coefficient of the obtained complex for C major characterized by 4-chords, its topology appears to be a *toroid* (the volume bounded by a torus) which is definitively different from a Möbius strip (*e.g.*, the torus is orientable and the Möbius strip is not.)

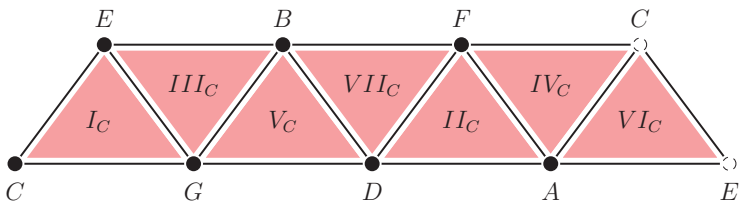


FIGURE 7
Simplicial representation of C major tonality resulting in a Möbius strip.

4.4 Computer Aided Analysis and Composition using Self-Assembled Spaces

In the context of *Music Set Theory* chords are studied from an algebraic point of view [1]. They are then classified as orbits under the action of some group of transformations. We are here interested in the classification induced by the action of the dihedral group.

Chord Classes and Generalized Tonnetze. The dihedral group D_{12} allows to gather chords up to transposition and inversion. While transposition T_i consists of the action of some interval on the notes of a chord as mentioned above, inversion S considers the inverse of the notes. As an example, $(T_{M3} \circ S)\{C, E, G\} = \{-C + M3, -E + M3, -G + M3\} = \{C, E, A\}$. This action defines 224 equivalent classes of chords.

Using the self-assembly process, we propose to represent each chord class by a simplicial complex. The topology of these classes differ widely from one to another depending on the size of the considered chords and their interval contents. A complete mathematical study of the topologies for 3-chords can be found in [6]. Figure 8 illustrates *unfolded* representations of the self-assembled structures 1-skeleton for the classes of $\{C, E, G\}$ and $\{C, E, G, B\flat\}$.

A remarkable fact is that the 1-skeletons of chord class spaces are particular graphs known in music theory as *Generalized Tonnetze*. Tonnetz stands

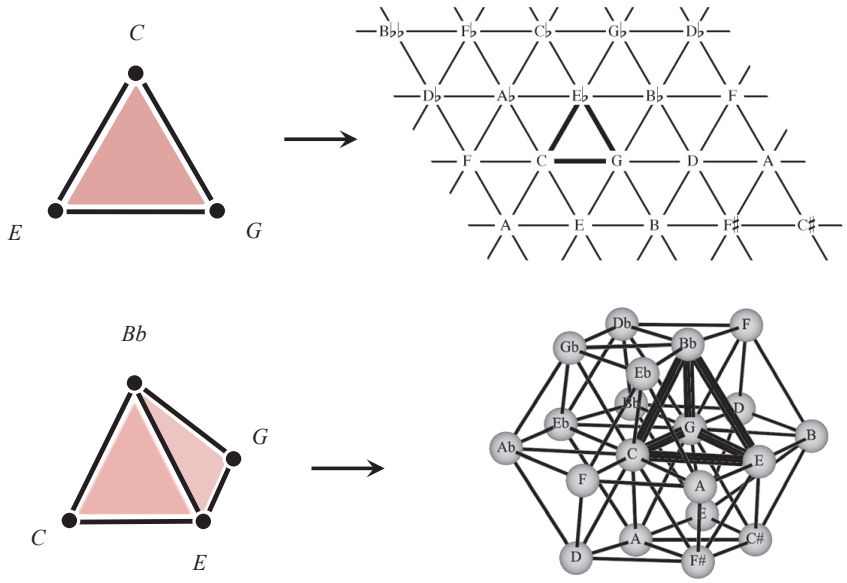


FIGURE 8
Assembly of chords equivalent up to transposition and inversion to C Major (C, E, G) (top) and to $C7$ $(C, E, G, B\flat)$ (bottom).

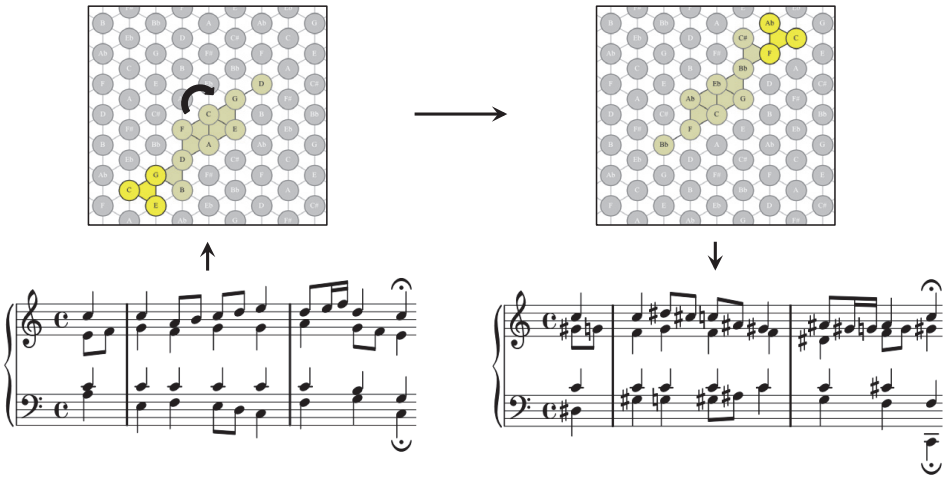


FIGURE 9
A discrete π rotation is applied on a trajectory representing the first measures of J.-S.Bach's choral BWV 256. The underlying space is the chord class complex built from the assembly of minor and major chords.

for tone network and consists in a regular graph labeled by pitch classes and generated by the action of a subset of intervals. Nevertheless higher dimensional cells of chord class spaces provide additional informations not present in original tonnetze and open new analysis and composition possibilities.

Transformations on musical trajectories. A musical sequence can be represented as a moving object evolving in a space, as the ones illustrated on Figure 8. The static representation of this evolution is a trajectory inside the complex, which consists of sub-complexes representing successive played notes. Figure 9 illustrates two trajectories in a chord class space. Discrete counterparts of some euclidean transformations can be applied on such trajectories. Figure 9 illustrates a discrete π rotation applied on a trajectory representing the first measures of J.-S. Bach's choral BWV 256. The underlying space is the chord class complex built from the assembly of minor and major chords. Moreover a spatial transformation can be applied on the underlying space, rather than on the path. These transformations occur on the labels of the elements only, which means that the topological structure of the underlying space stays unchanged. This property ensures that the aspect of the trajectory is conserved.

Some of these spatial transformations have a musical interpretation. For example, a translation leads to a transposition or a modal transposition, depending on the underlying space. A π rotation leads to an musical inversion. Other available transformations lead to alternative musical operations.

Audio results of rotations, translations and transformations of the underlying space, applied to some pieces from various composers (J.-S. Bach, W.A. Mozart, M. Babbit, The Beatles,. . .) are available in MIDI format at http://www.lacl.fr/ lbigo/aisb13#transformations. Example 5 of the online page corresponds to the transformation illustrated on Figure 9.

HexaChord and PaperTonnetz Software. The previous theoretical considerations have been implemented in two original software based on the notions of generalized Tonnetz and chord class: HexaChord [†] and PaperTonnetz [‡] .

HexaChord is a computer-aided music analysis environment based on the previous spatial representation of chord classes. Beyond the simple visualization of the different 12 2-dimensional simplicial complexes, HexaChord provides musicologists with three main functionalities on imported midi files:

1. HexaChord computes automatically the best Tonnetze for representing some piece; it is based on the computation of a *compliance measure* defined at general level on cellular complexes.
2. Giving a particular Tonnetze, HexaChord is able to produce a trajectory from a piece to show interesting geometric properties.
3. Harmonization by spatial criteria is available by adding an extra voice to a set of voices (*e.g.*, a choral) to maximize compliance measurement.

PaperTonnetz [3] is a tool that lets musicians explore and compose music with Tonnetz representations by making gestures on interactive paper. It allows composers to discover, improvise and assemble musical sequences in a Tonnetz by creating replayable patterns that represent pitch sequences and/or chords. Figure 10 shows GUI of PaperTonnetz and HexaChord.

5 CONCLUSION

A notable contribution of this paper is to show that the same generic self-assembly process can be used to build abstract spaces representing various well-known musical objects: the all-interval series, the simplicial representations of chord sets and the associated Tonnetze.

The generic self-assembly process has been implemented in MGS [¶] providing a formal and very concise specification. The corresponding MGS transformation has been effectively used to compute and enumerate these

[†] http://vimeo.com/38102171
[‡] http://vimeo.com/40072179
[¶] http://mgs.spatial-computing.org

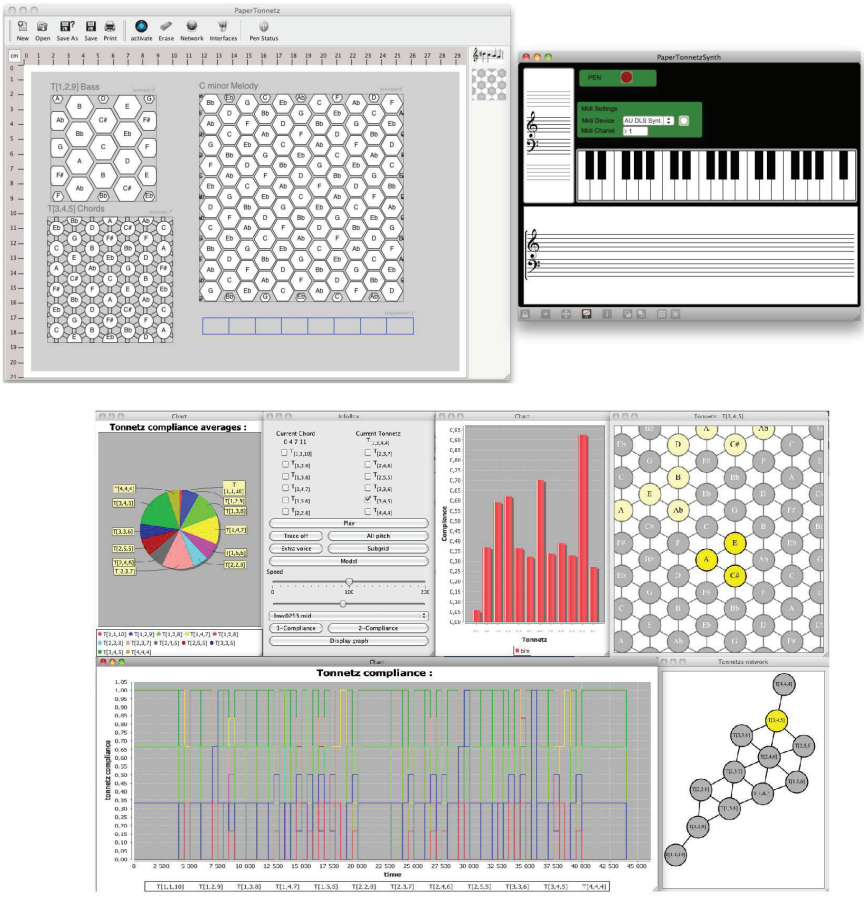


FIGURE 10
Graphical User Interface of PaperTonnetz (top) and HexaChord (bottom).

spaces. Then, topological invariants can be computed to investigate and classify them, especially when they have a high dimension.

Works presented here are related to the unconventional computing field in many aspects. First of all, the different computations presented here are specified using topological rewriting, an extension of term rewriting to abstract cellular complexes. Such extension subsumes chemical computing *à la* Gamma (as commutative-associative rewriting), Lindenmayer systems (as string rewriting) and cellular automata (as a kind of constrained array rewriting). Secondly, the algorithms used are directly inspired by mechanisms investigated in the field of unconventional computing. These mechanisms are used as a very effective heuristic to devise new representations of well known musical objects and processes. For instance, we use self-assembly to build a space associated with a set of chords and we use computations in space to represent and extend musical transformations. Last but not least, from a musical

perspective the transformations presented here are highly non conventional. As a matter of fact, the topological framework has been used only very recently to formalize musical notions, while the standard approach relies solely on algebraic structure, *e.g.*, for pitch classes in Set Theory. The topological framework renews our point of view on musical problems and suggests some alternative musical transformations. It also suggests new generative processes to produce music, an area not covered in this paper.

We believe that this preliminary work shows the interest of an expressive and generic framework making possible the systematic building and processing of abstract spaces that appear in musical analysis and theory. Our framework is based on spatial notions developed and studied in algebraic topology, and then amenable to a computer implementation due to their algebraic nature. Initially developed for the modeling and the simulation of dynamical systems, it appears well suited for the musicologist.

Designing original interesting spaces is a first step work. We are currently working on unconventional approaches (*e.g.*, cellular automata, random walk algorithms, etc.) for taking advantage of these spaces for compositional purposes.

ACKNOWLEDGMENTS

The authors are very grateful to C. Agon, M. Andreatta, G. Assayag, J.-L. Giavitto from the REPMUS team at IRCAM and to O. Michel from the LACL at University of Paris-Est for fruitful discussions. This research is supported in part by the IRCAM and the University Paris Est-Créteil Val de Marne.

REFERENCES

[1] M. Andreatta and C. Agon. (2003). Implementing algebraic methods in openmusic. In *Proceedings of the International Computer Music Conference, Singaphore*.

[2] J. Beal, S. Dulman, K. Usbeck, M. Viroli, and N. Correll. (2012). Organizing the aggregate: Languages for spatial computing. *arXiv preprint arXiv:1202.5509*.

[3] L. Bigo, J. Garcia, A. Spicher, W.E. Mackay, *et al*. (2012). Papertonnetz: Music composition with interactive paper. In *Sound and Music Computing*.

[4] L. Bigo, J.L. Giavitto, and A. Spicher. (2011). Building topological spaces for musical objects. *Mathematics and Computation in Music*, pages 13–28.

[5] C. Callender, I. Quinn, and D. Tymoczko. (2008). Generalized voice-leading spaces. *Science*, 320(5874):346.

[6] M.J. Catanzaro. (2011). Generalized tonnetze. *Journal of Mathematics and Music*, 5(2):117–139.

[7] André De Hon, Jean-Louis Giavitto, and Frédéric Gruau, editors. (3-8 september 2006). *Computing Media and Languages for Space-Oriented Computation*, number 06361 in Dagsthul Seminar Proceedings. Dagsthul, http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=2006361.

[8] I. P. Gent, T. Walsh, I. Hnich, and I. Miguel, (accessed in january 2011). CSPLib: a problem library for constraints. web page at http://www.csplib.org.

[9] J.-L. Giavitto and O. Michel. (2003). Modeling the topological organization of cellular processes. *BioSystems*, 70(2):149–163.

[10] Jean-Louis Giavitto. (2003). Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In *Rewriting Technics and Applications (RTA'03)*, volume 2706 of *Lecture Notes in Computer Science*, pages 208–233, Valencia. Springer.

[11] Jean-Louis Giavitto and Antoine Spicher. (2008). *Systems Self-Assembly: multidisciplinary snapshots*, chapter Simulation of self-assembly processes using abstract reduction systems, pages 199–223. Elsevier. doi:10.1016/S1571-0831(07)00009-3.

[12] J.L. Giavitto, O. Michel, and A. Spicher. (2011). Interaction based simulation of dynamical system with a dynamical structure (ds) 2 in mgs. In *Proceedings of the 2011 Summer Computer Simulation Conference*, pages 99–106. Society for Modeling & Simulation International.

[13] Franck Jedrzejewski. (2006). *Mathematical Theory of Music*. Delatour.

[14] G. Mazzola *et al*. (2002). *The topos of music: geometric logic of concepts, theory, and performance*. Birkhäuser.

[15] Olivier Michel, Antoine Spicher, and Jean-Louis Giavitto. (December 2009). Rule-based programming for integrative biological modeling – application to the modeling of the λ phage genetic switch. *Natural Computing*, 8(4):865–889.

[16] Robert Morris and Daniel Starr. (1974). The structure of all-interval series. *Journal of Music Theory*, 18(2):pp. 364–389.

[17] James Munkres. (1984). *Elements of Algebraic Topology*. Addison-Wesley.

[18] Thorvald Otterström. (1935). *A theory of Modulation*. Da Capo press, Inc.

[19] André Riotte and Marcel Mesnage. (2006). *Formalismes et modèles musicaux*. Delatour.

[20] David Schiff. (1983). *The Music of Elliott Carter*. Faber and Faber.

[21] SCW, (accessed in january 2011). the "Spatial Computing Workshops" series and related events. List on the "Spatial Computing Home Page" http://www.spatial-computing.org/doku.php?id=events:start.

[22] Antoine Spicher, Olivier Michel, and Jean-Louis Giavitto. (September 2010). Declarative mesh subdivision using topological rewriting in mgs. In *Int. Conf. on Graph Transformations (ICGT) 2010*, volume 6372 of *LNCS*, pages 298–313.

[23] C. Truchet and P. Codognet. (2004). Musical constraint satisfaction problems solved with adaptive search. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 8:633–640.

[24] A.W. Tucker. (1933). An abstract approach to manifolds. *Annals of Mathematics*, 34(2):191–243.

[25] D. Tymoczko. (2006). The geometry of musical chords. *Science*, 313(5783):72.