Building Topological Spaces for Musical Objects

Louis Bigo, Jean-Louis Giavitto, Antoine Spicher

Building Topological Spaces for Musical Objects

Louis Bigo[1,2], Jean-Louis Giavitto[2], and Antoine Spicher[1]

[1] LACL/Université Paris-Est Creteil
louis.bigo@lacl.fr, antoine.spicher@u-pec.fr
[2] UMR CNRS STMS 9912/IRCAM
jean-louis.giavitto@ircam.fr

Abstract. The development of spatial representations of musical objects allows for a reformulation of algorithmic problems arising in musical theory, fosters novel classifications and provides new computational tools. In this paper, we show how a topological representation for n-note chords associated with the degrees of the diatonic scale and for the All-Interval Series (AIS) can be automatically built using MGS, a rule-based spatial programming language. Then, we suggest a new categorization for AIS based on their spatial construction.

Keywords: Spatial computing, algebraic topology, All-Interval Series, n-chords and diatonic scale, self-assembly, n-Hamiltonian path.

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 [22,3,2] or build topological spaces embedding musical relationships in their neighborhood relationships [10].

In this paper we will follow this line of research by using tools developed in *spatial computing*. Spatial computing is an emergent domain in computer science that enlightens the notion of space in computations either as a *resource* or a *result* or a *constraint* [4,17]. Spatial computing has proven to be a fruitful paradigm for the (re-)design of algorithms tackling problems embedded in space or having a spatial extension.

In the following, we propose two studies of paradigmatic theoretical music problems from a spatial computing perspective. This paper is organized as follows. Section 2 provides a brief introduction to MGS, a domain specific programming language designed to investigate the spatial computing approach. Subsection 2.3 illustrates the MGS concepts on a self-assembly algorithm for the generic computation of a topological representation of chord series initially proposed by Guérino Mazzola. In section 3 we propose a spatial representation of the constraints defining an All-Interval Series (AIS). This representation enables us to enumerate the AIS and to classify them from a topological perspective. The paper ends with a conclusion and a discussion about future works.

2 Presentation of the MGS programming language

MGS is an experimental domain specific language dedicated to spatial computing, see [7,6]. MGS concepts are based on well established notions in algebraic topology [13] and relies on the use of rules to compute declaratively with spatial data structures.

In MGS, all data structures are unified under the notion of *topological collection*: an *abstract combinatorial complex* (ACC) labeled with arbitrary values. The ACC builds a space in a combinatorial way through more simple objects called *cells* [21]. It acts as a container and the values as the elements of the data structure. *Transformations* of topological collections are defined by rewriting rules [19] specifying replacement of sub-collections that can be recursively performed to build new spaces.

2.1 Topological Collections

An *abstract combinatorial complex* $K = (\mathsf{C}, \prec, [\cdot])$ is a set C of abstract elements, called *cells*, 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 0 corresponds to a point, a 1-dimensional cell corresponds to a line (an edge), a cell of dimension 2 is a surface (*e.g.* a polygon), etc. A cell of dimension p is called a p-cell. For example, a graph is an ACC built only with 0- and 1-cells. An other example is pictured in Fig. 1.

In the context of this paper, we write $\partial_K c$ for the sub-ACC made of the cells of K lower than c for the relation $\prec$: $\partial_K c = (\mathsf{C}', \prec \cap\ \mathsf{C}' \times \mathsf{C}', [\cdot])$ where $\mathsf{C}' = \{c' \,|\, c' \prec c\}$. This ACC is called the *boundary* of c. The *faces* of a p-cell c are the $(p-1)$-cells c' of $\partial_K c$ and we write $c > c'$ or $c' < c$ and c' is called a *coface* of c.

Two cells c and c' are *q-neighbor* either if they have a common border of dimension q or if they are in the boundary of a q-cell (of higher dimension). If

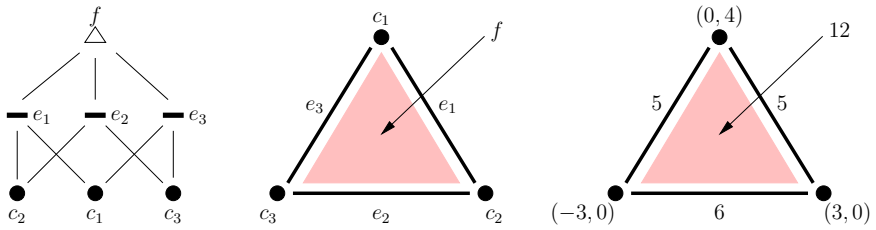


Fig. 1. On the left, the Hasse diagram of boundary relationship of the ACC given in the middle: it is composed of three 0-cells (c_1, c_2, c_3), of three 1-cells (e_1, e_2, e_3) and of a single 2-cells (f). The three edges are the faces of f, and therefore f is a common coface of e_1, e_2 and e_3. On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

the two cells are of dimension p, we say that they are (p,q)-*neighbor*. A (p,q)-*path* is a sequence of p-cells such that two consecutive cells are q-*neighbor*. For example, the usual notion of path in a graph (a sequence of vertexes such that from each of its vertexes there is an edge to the next vertex in the sequence) corresponds to the notion of $(0,1)$-path.

Topological Collections. A topological collection C is a function that associates a value with a cell in an ACC, see Fig. 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

The mechanics of rewriting systems are familiar to anyone who has done arithmetic simplifications: an arithmetic expression can be simplified by repeatedly replacing parts of the term (*subterms*) with other subterms. For example, $\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} \Rightarrow \frac{1}{3}\cdot\frac{3}{4} \Rightarrow \frac{1}{4}$. The rule that is applied here is: $\frac{M}{N}\cdot\frac{N}{P} \Rightarrow \frac{M}{P}$, where M, N and P are pattern variables representing arbitrary non-null numbers. A transformation generalizes this process to topological collections.

Topological collections are transformed using sets of rules called *transformations*. 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 by the evaluation of `expression`. There exist several ways to control the application of a set of rules on a collection but these details are not necessary for the comprehension of the work presented here. A formal specification of topological rewriting is given in [19]. We sketch here only the specification of patterns.

A *pattern variable* specifies a cell to be matched in the topological collection together with some (optional) guard. For example 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. There are three composition operators:

1. The composition denoted by a simple juxtaposition (e.g., "`x y`") does not constraint the arguments of the composition.
2. When two pattern variables are composed using a comma (e.g., "`x, y`"), it means that the cells matched by `x` and `y` must be p-neighbors. The default

value for p is 1 and can be explicitly specified during the application of the transformation if needed.

3. The last composition operator corresponds to the face operator: a pattern "`x < y`" (resp. "`x > y`") matches two cells $c_{\mathtt{x}}$ and $c_{\mathtt{y}}$ such that $c_{\mathtt{x}} < c_{\mathtt{y}}$ (resp. $c_{\mathtt{x}} > c_{\mathtt{y}}$).

Patterns are *linear*: two distinct pattern variables always refer to two distinct cells.

2.3 Self-Assembly

In this subsection, we illustrate the notions of topological collection and transformation with a self-assembly mechanism used to compute a spatial representation of a chord set. Guérino Mazzola presents in [11] a topological representation of the seven trichords associated with the degrees of the diatonic scale (for example C major) which appears to be a Möbius strip. We propose here a generic self-assembly process that achieves the same construction for any set of chords.

Spatial Representation of a Chord. We represents a n-note chord by a $(n-1)$-simplex. A *simplex* is a p-cell that has exactly $p+1$ faces. For instance, a bounded line is a simplex but a cube is not (a cube is a 3-cell but has 6 faces). Simplexes are often represented geometrically as the convex hull of their vertexes as shown in Fig. 2 for p-simplexes with $p \in \{0,1,2,3\}$. In the simplicial representation of chord, a 0-cell represents a single note.

The first step of a spatial representation of a set of chords, consists in giving a simplicial representation (as presented on Fig. 2) of each chord. As an example, for the spatial representation of the C major tonality, the seven degrees

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

are associated with seven 2-simplexes. The complex associated with a chord is thus represented by one 2-cell (a triangular surface) whose boundary is composed of three 1-cells (edges) and three 0-cells (vertexes) respectively associated with three 2-note chords and a single note. See Fig. 2.

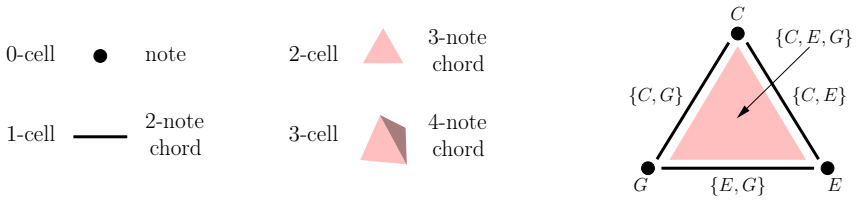


Fig. 2. A chord represented as a simplex. The complex on the right corresponds to first degree I_C of the C major tonality and all 2-note chords and notes included in it.

The Space of Chords. To gather in the same ACC all the considered chord simplexes and their respective boundaries, we must identify the various simplexes representing the same chord. For instance, merging chords I_C and III_C requires us to identify three elements: the vertexes E and G, and the two edges $\{E, G\}$ as shown on Fig. 3.

We propose to unify these elements by using a self-assembly growing process [8] based on the identification of the cells boundaries. This operation is not elementary because the identification must occur at every dimensions. A simple way to compute the identification is to iteratively apply, until a fixed point is reached, the merge of topological cells that exactly have the same faces. The corresponding topological surgery can be expressed in the MGS syntax as follows:

```
transformation identification = {
    s1 s2 / (s1==s2 & faces(s1)==faces(s2))
    => let c = new_cell (dim s1)
                        (faces s1)
                        (union (cofaces s1) (cofaces s2))
       in  s1·c
}
```

The primitive `new_cell` $p\ f\ cf$ returns a new p-cell with faces f and cofaces cf. The rule specifies that two elements `s1` and `s2`, having the same label and the same faces in their boundaries, merge into a new element `c` (whose cofaces are the union of the cofaces of `s1` and `s2`) labeled by `s1` (which is also the label of `s2`).

In Fig. 3, the transformation `identification` is called twice. At the first application (from the left complex to the middle), vertexes are identified. The two topological operations are made in parallel. At the second application (from the complex in the middle to the right), the two edges from E to G that share the same boundary, are merged. The cofaces of the resulting edge are the 2-simplexes I_C and III_C corresponding to the union of the cofaces of the merged edges. Finally (on the right), no more merge operation can take place and the fixed point is reached.

As expected, the fixed point application of `identification` to the triadic chords I_C, ..., VII_C builds the Möbius strip given in Figure 4. Notice that in [11], the dual complex is described: in this complex, a vertex represents a

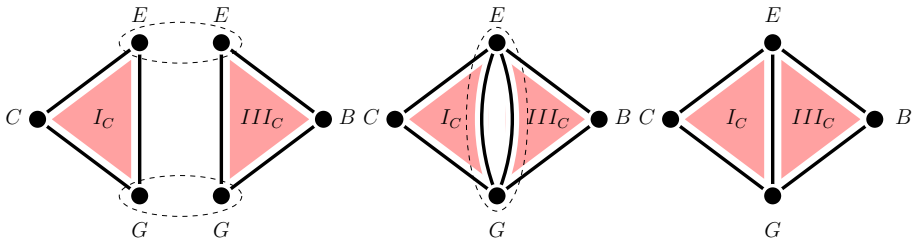


Fig. 3. Identification of boundaries.

3-note chord, an edge links two vertexes that share two notes and a face corresponds to a note common to three vertexes. Obviously, we can adapt the construction presented here to self-assemble this complex but our representation makes the dimension of a note independent of the number of notes in the considered chords.

Higher Dimensional Spaces. The previous transformation is generic with respect to the dimension of the simplexes. So, it can be applied without modification for the p-note chord for any p. For example, the geometrical representation of a four-note chord is a 3-simplex, that is a tetrahedron (for example $I_C = \{C, E, G, B\}$). The elaboration of the cellular complex associated with the 4-note chords of the seven degrees is difficult by hand. Thanks to the dimension-free definition of the transformation `identification`, the cellular complex is automatically computed by MGS. After computing the Euler characteristic and the orientability coefficient of the complex, its topology appears to be a *toroid* (the volume bounded by a torus). The one dimensional boundary of the Möbius strip exhibits the cycle of fifths whilst the two dimensional boundary of the toroid exhibits a surface S composed of triangles representing 3-note chords. It is easier to interpret the dual D of this surface: each vertex in S corresponds to a polygonal surface in D and conversely each surface in S correspond to a vertex in D (edges remain edges). The dual D can be visualized as an hexagonal lattice interpreted as a kind of *tonnetz*. This lattice is generated by the intervals of third, fifth and seventh, cf. Fig. 5. These intervals can be perfect or diminished (for the fifths) and minor or major (for thirds and sevenths) in order to generate only notes in the C major scale. This lattice includes the part of the circle of fifths composed by the notes of the C major tonality.

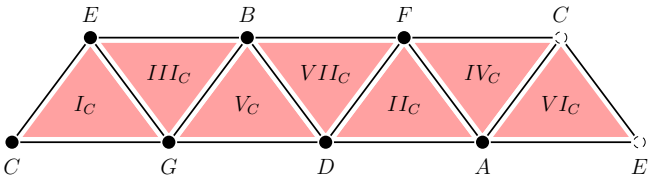


Fig. 4. Representation of tonality C major. The edge and the vertexes at one end must be merged with the edge and the vertexes with the same label at the other end. The resulting surface is a Möbius strip.

3 All-Interval Series

In this section we illustrate a general methodology of spatial computing, using the enumeration and classification of *All-Interval Series* (AIS) as an example. We first make a short presentation of AIS. Then we propose a spatial specification of AIS. This specification is further used to characterize some noticeable AIS and to classify them.

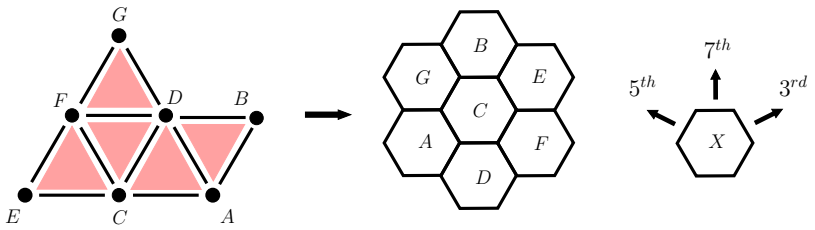
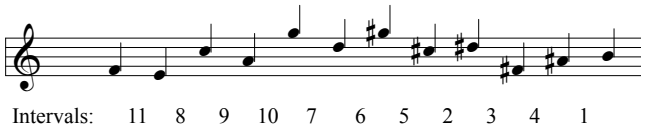


Fig. 5. The toroid boundary is the surface specified by the triangular lattice S (unfolded on the left). Its dual is the hexagonal latice D (on the center) where notes are organized following the intervals of fifth, seven and third (on the right).

3.1 Presentation of AIS

The enumeration and the classification of AIS is a widely known problem in the computing music community. An AIS is a twelve-tone series including eleven different intervals reduced in $\mathbb{Z}_{12}$. Such series contain many notable properties [12]. One of them is that the first and the last notes are always separated by a tritone interval. One of the most known 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. Different computing approaches, increasingly optimized, have been used to enumerate the totality of the AIS. One of the first enumeration was done by André Riotte [15] with the help of a FORTRAN program. This enumeration problem has quickly become a classical problem in Constraint Programming [20] and is now part of the 50 problems of the CSPLib [5]. Some previous works have been based on the enumeration of the All-Interval Chords, which is a similar problem [14].

Beyond the enumeration of the 3 856 AIS, composers and music analysts have been interested in finding pertinent 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 sub-sequence corresponding to the notes of particular scales or chords [15]. We will propose an example motivated by a similar goal in the next section. Elliot Carter investigated a classification enumerating all AIS containing in sequence the complete set of notes included in the All-Triad Hexachord (this 6-note chord is the only one containing the twelve possible triads) [16]. We can also mention an original classification from Franck Jedrzejewski based on knot theory [9].

3.2 Enumeration of All-Interval Series

We propose a new approach for the old problem of AIS enumeration. Our solution is based on the construction of a topological space adapted for their combinatorial analysis.

AIS can be seen as objects that inhabit a specific and well designed abstract space. Let's respectively call this space and the objects the *support space* and the *solutions* of the computation. The computation of the AIS consists in finding all the solutions in the support space. Space is considered as a *constraint* to guide the search: the more structured the space, the more efficient the computation. Thus, the challenge lies in the construction of a relevant support space together with an efficient search algorithms to find the solutions in the support space.

The notion of *search space* in the domain of search algorithms corresponds to the special case of support space where the possible solutions are the points of this space. In our example, instead of building a search space of $12! \simeq 479.10^6$ points and looking for the relevant solutions, we will build a much smaller support space composed of 12 points and we will look for paths in this space exhibiting some relevant properties: here, the paths associated with AIS are *Hamiltonian*.

A first naive method, corresponding to a brute force search, is given to explain our approach. Finally by adding more spatial structure to the support space, we increase the efficiency of the computation.

A Brute Force Approach. A brute force approach starts by considering the set of the twelve notes, and consists in computing all the possible permutations and keeping those having exactly eleven different intervals.

The spatial expression of this algorithm is straightforward. The support space of the computation is here the set of the twelve notes. From a spatial point of view, a set is a topological collection corresponding to a complete graph: each set element is associated with a vertex labelled by the element itself. The complete graph topology, meaning that there is an edge between each pair of vertexes, specifies that there is no predefined order between the elements. Fig. 6 presents the support space we consider for the computation of the AIS.

In this space, a permutation is an Hamiltonian path where each note appears only once. Using the definitions given section 2, these paths exactly correspond to the $(0,1)$-paths of maximal length: the elements are the 0-cells that have to be neighbor by dimension 1 (that means linked by a 1-cell). An example of a $(0,1)$-path is given in bold on Fig. 6. Such path is easily expressed using an MGS pattern:

```
n0, n2, ..., n11 / ais(n0,...,n11)
```

In this pattern, the comma stands for the $(0,1)$-neighborhood. The additional guard `ais(n0,...,n11)` checks if the Hamiltonian path matched by the `n`i is an AIS or not: the predicate `ais` holds if its arguments correspond exactly to 11 distinct intervals. Thus, this pattern specifies exactly the solutions of our computation.

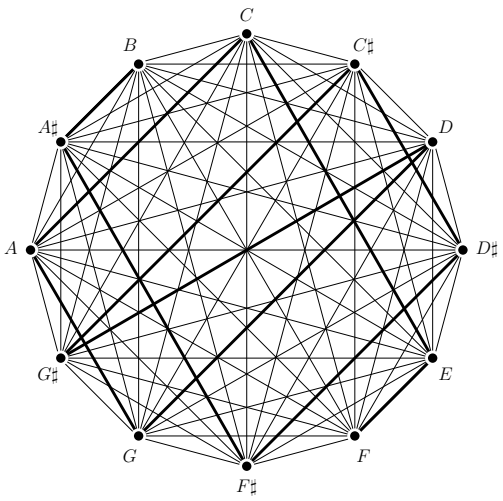


Fig. 6. Spatial representation of Alban Berg's AIS (in bold in the complete graph).

Optimizing a spatial search algorithm can be done at the level of the solution specification and at the level of the support space definition. We propose in the following both optimizations.

Optimization of the Solution Specification. One can easily notice that the largest part of the computations involved by the brute force approach is useless. Indeed, it is possible in most of the cases to detect that a series is not all-interval before having determined the twelve notes. For example, if the interval between `n0` and `n1` is the same as the interval between `n1` and `n2`, there is no need to look further. In other words, predicate `ais` can be distributed along the path specification in the pattern.

Suppose that the edges of the graph pictured in Fig. 6 are labeled by the corresponding intervals (*e.g.* the edge between E and G is labeled by 3). The following pattern distributes the evaluation of the predicate `ais` along the matching of the path:

```
n0 < i1  > n1
   < i2  > n2  / (i2!=i1)
   < i3  > n3  / (i3!=i1) / (i3!=i2)
   ...
   < i11 > n11 / (i11!=i1) / ... / (i11!=i10)
```

This pattern is quite similar to the previous one. One of the major differences stands in the use of `< i`p `>` instead of the comma to express the neighborhood. The pattern variable `i`p corresponds to the interval between the notes `n`p' and `n`p with $p' = p - 1$. Assuming that the edges are labelled by the corresponding

interval, the path specifies an AIS if all the labels `i`p are distinct[3]. This property is ensured by the guards (`i`p`!=i`q) that are distributed along the pattern as required by the optimization. The search algorithm induced by this pattern corresponds to the algorithm of the FORTRAN program given in [12]. This new pattern speeds-up the computation of the AIS by a factor of 30.

Optimization of the Support space. There are two different kinds of constraint in the solution specification: *spatial constraints* and *logical constraints*. Spatial constraints specify which sub-parts of the support space are structural candidates for being solutions without taking labels into account. On the other hand, logical constraints are used to determine the solutions among the structural candidates by checking some formulas on labels. The two previous patterns are spatially equivalent. In fact, the previous optimization does not decrease the number of structural candidates but only reduces the number of visited candidates from 12! to about 9.10^6 during the computation by avoiding the whole construction of wrongly started paths.

From a spatial point of view, it is more interesting to optimize the search by reducing the number of structural candidates. Ideally, we are looking for a totally spatial solution specification (without any logical constraint).

In our example, the proliferation of structural candidates comes from the lack of spatial distinction between intervals. Indeed, each interval is instantiated several times: for example, the semitone corresponds to the edge between C and $C\sharp$, to the edge between $C\sharp$ and D, etc. Therefore, we propose, for each interval i, to add in the support space, a 2-cell whose faces are exactly the 1-cells incident to two notes separated by the interval i. This cell represents the class of all the intervals i. Fig. 7 illustrates the boundary relationships defined for the classes of the fourth and the minor third. The 2-cells are filled in light gray. Notice that the topologies of the classes differs: in the given two examples, the minor third exhibits two holes while the fourth has no hole.

This new space is easily built following the method exposed in section 2.3. In this space, the specification of the AIS is completely structural. We are looking for a path, made of 0, 1 and 2-cells, which is 0-Hamiltonian and also 2-*Hamiltonian*[4]:

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

Fig. 8 illustrates the instantiation of this path on the five first elements of Alban Berg's AIS. It is easy to see on this figure that the spatial constraint consists in

[3] We do not enter in the technical details of the correct handling of the edges orientation. We assume that the pattern `c < e > c'` means that `c` (resp. `c'`) is negatively (resp. positively) oriented with respect to `e`.

[4] A path in a graph which visits every edge exactly once is called *Eulerian*. However there is no general name for path in a complex which visits the p-cells exactly once. We name such path a p-Hamiltonian path.

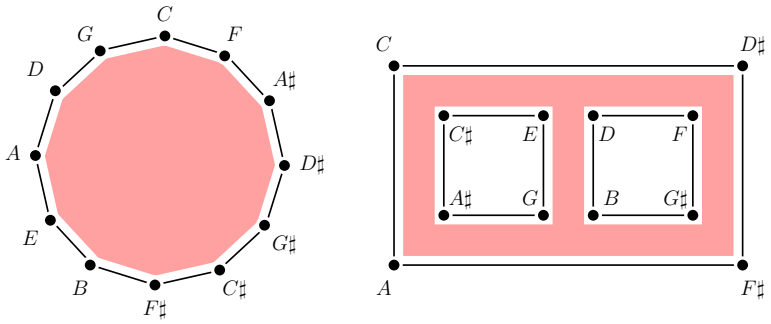


Fig. 7. Spatial representation of the 2-cells (and their boundaries) of the interval classes associated with the fourth (on the left) and with the minor third (on the right)

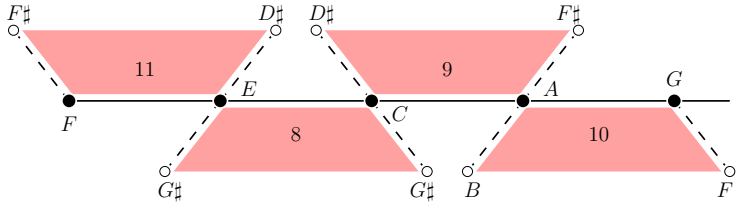


Fig. 8. Spatial representation of the five first notes of Alban Berg's AIS

matching an additional 2-cell. More precisely, two consecutive notes $\mathtt{n}p'$ and $\mathtt{n}p$ with $p' = (p-1)$ have to be $(0,1)$-neighbors by $\mathtt{i}p$ and $(0,2)$-neighbors by $\mathtt{I}p$ and such that cells $\mathtt{I}p$ and $\mathtt{i}p$ are incident.

Finally, the AIS computation time decreases by a factor of 4 with this solution specification applied on the original support space extended with 2-cell intervals. The speed-up obtained by tailoring the support space w.r.t. the brute force approach is greater than 120.

3.3 A Spatial Classification of AIS

By definition, each AIS visits only one 1-cell in the boundary of each 2-cell. A natural idea to classify the AIS is then to look further in the structure of the 2-cells boundary. As we can see on Fig. 7, the boundary of a 2-cell is composed of a variable number of disconnected cycles (here a cycle is a closed $(0,1)$-path). Each cycle is one of the orbits of the action of the corresponding interval on the set of notes. For an interval i, it is well known that the number of orbits, hence cycles in the boundary of the associated 2-cell, is $d_i = \gcd(i, 12)$. For instance, there are $\gcd(3, 12) = 3$ cycles in the boundary of the minor third class 2-cell:

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

These cycles can be uniquely identified by an integer between 0 and $d_i - 1$ corresponding to the least note of the orbit (with the usual identification between pitch classes and numbers: $C = 0, C\sharp = 1$, etc.).

We can associate with an AIS a vector $\mathbf{V} = (v_1, \ldots, v_i, \ldots, v_{11})$ of 11 numbers giving, for each interval class, the visited cycle. This vector is called the *cyclic vector* of the AIS. Beware that the i^{th} element in this vector is associated with the interval i which does not usually coincide with the i^{th} interval in the AIS. For example, the cyclic vector $\mathbf{V}_B$ of the AIS in the *Lyric Suite* is

interval class i	1	2	3	4	5	6	7	8	9	10	11
d_i	1	2	3	4	1	6	1	4	3	2	1
$\mathbf{V}_B$	0	1	0	2	0	2	0	0	0	1	0

A *cluster* is the set of AIS sharing the same cyclic vector.

Construction of a Cyclic Vector Compliant with a Given Scale. As previously mentioned, some composers and analysts are interested in building some AIS including a sub-sequence of notes having a particular meaning [15,16].

For instance, André Riotte has analyzed AIS containing a sub-series of notes belonging to a particular scale. This problem can be formalized in various ways. Based on our representation, we propose the following formulation: given a scale S containing some whole orbits for some intervals, compute the AIS whose cyclic vectors includes these orbits. In other words, the two consecutive notes in the AIS separated by such an interval belong to S.

For example, the harmonic C minor scale $C\ D\ E\flat\ F\ G\ A\flat\ B$ contains two orbits $(D - F - A\flat - B)$ and $(E\flat - G - B)$ for the interval of minor third and of major third (and the two corresponding retrograde orbits for major sixth and for minor sixth). Following the definition of the cyclic vector, these two orbits are respectively identified by the integers $v_3 = 2$ and $v_4 = 3$. In the same way, we arbitrarily propose to use, for the tritone class interval, the cycle $(F - B)$ whose notes are included in the scale too. This choice reaches to define $v_6 = 5$. Interval classes of minor second ($i = 1$), perfect fourth ($i = 5$), perfect fifth ($i = 7$) and major seventh ($i = 11$) correspond for each to a single orbit (identified by the integer 0). We can summarize the choice of the orbits for each interval in the following table:

interval class i	1	2	3	4	5	6	7	8	9	10	11
d_i	1	2	3	4	1	6	1	4	3	2	1
$\mathbf{V}$	0	$*$	2	3	0	5	0	$*$	$*$	$*$	0

Among the two possible orbits of the major second class interval, we choose the cycle $(C\sharp - D\sharp - F - G - A - B)$ because it contains four of the seven notes of the scale, i.e. $v_2 = 1$. A check of the remaining possible combinations for the cyclic vector, gives the following vectors:

$\mathbf{V_1}$	0	1	2	3	0	5	0	0	0	0	0
$\mathbf{V_2}$	0	1	2	3	0	5	0	0	1	0	0
$\mathbf{V_3}$	0	1	2	3	0	5	0	2	0	0	0
$\mathbf{V_4}$	0	1	2	3	0	5	0	2	1	0	0

Each vector is associated with a different non-empty set of AIS. Because of the constraints on the orbits we have imposed for some intervals, we know that all these AIS include several times in their structure two consecutive notes included in the harmonic C minor scale. A research of AIS containing the longer subsequence of notes belonging to the scale reaches to the following series associated with the vector $\mathbf{V_2}$:

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

As we can observe, this AIS contains in sequence the seven notes of the harmonic C minor scale.

Geometry of the Clusters. There are about $\prod_i d_i = 3\,456$ possible clusters where the 46 272 AIS are scattered (some clusters are empty). We introduce here some preliminary idea to study how they are related with each other in the light of the standard algebraic operations sending an AIS to another one [12]. Given an AIS $(n_0, \ldots, n_j, \ldots, n_{11})$ with corresponding cyclic vector $\mathbf{V} = (v_1, \ldots, v_i, \ldots, v_{11})$, the application of an operation φ computes an AIS $(n'_0, \ldots, n'_j, \ldots, n'_{11})$ with corresponding cyclic vector $\mathbf{V}' = (v'_1, \ldots, v'_i, \ldots, v'_{11})$ defined by:

- for the *transposition* $\varphi = T_b$

$$n'_j \equiv n_j + b \mod 12 \qquad \text{and} \qquad v'_i \equiv v_i + b \mod d_i;$$

- for the *homothety* $\varphi = H_p$ (p relatively prime with 12)

$$n'_j \equiv p\, n_j \mod 12 \qquad \text{and} \qquad v'_i \equiv p\, v_{p^{-1}\, i} \mod d_i;$$

- for the *retrograde* $\varphi = R$

$$n'_j \equiv n_{-j-1} \mod 12 \qquad \text{and} \qquad v'_i \equiv v_{-i} \mod d_i;$$

- for the *circular shift* $\varphi = Q$ (w is the position of the tritone)

$$n'_j \equiv n_{j+w} \mod 12 \qquad \text{and} \qquad v'_i \equiv \begin{cases} n_0 & \mod d_i \quad \text{if } i = 6 \\ v_i & \mod d_i \quad \text{otherwise} \end{cases}.$$

We aim at simplifying the study of the AIS distribution within the space of clusters by studying these operations.

Previous studies considered only the 3 856 *normalized* AIS (i.e., beginning with C); the others AIS can be reached by the 11 transpositions. Unfortunately this reduction is not relevant in the context of cluster classification which relies on the interval classes but not on the notes positions. In other words, the normalized AIS are not localized in a specific subset of clusters. Nevertheless, a normalized AIS always ends with $F\sharp$, so the circular shift of a normalized AIS produces an other AIS

$$Q(0, \ldots, n_w, n_{w+1}, \ldots, 6) = (n_{w+1}, \ldots, 6, 0, \ldots, n_w)$$

where the visited tritone class is $(C - F\sharp)$: thus, it belongs to a cluster with $v_6 = 0$. We can only consider the $\prod_{i \neq 6} d_i = 576$ clusters with $v_6 = 0$. Since these clusters also include the AIS of the form $(\dots, 0, 6, \dots)$, the symmetries induced by the retrograde R and the transposition T_6 can be used to reduce a step further the set of clusters. The 156 remaining clusters are completely identified and *exactly* include 3 856 different AIS (equivalent to the normalized one using R, T_6 and Q). Finally, the last operations, namely the homotheties, decrease this number to 72. This reduction process, as well as the proofs, are detailed in a companion technical report [18].

4 Conclusion and Future Work

This paper aims at presenting the first results in the application of the spatial computing paradigm to musical theory problems. The two problems illustrating our approach are the structure of the n-note chords of the diatonic scale and the classification of the AIS. They are hardly new, but the framework presented here, based on the topological notions supported by the MGS programming language, is generic. For instance, one can study systematically the simplicial complex associated with an n-note chord series for a wide range of n, even for $n > 4$ when it cannot be graphically visualized. The topological formalization of the AIS enumeration relies on the hamiltonicity of a path in a well-designed space, a notion already used elsewhere as a compositional tool (e.g., Robert Morris in *Hamiltonian Cycle: Saxophone*, Michael Winter in *Maximum Changes* or Giovanni Albini in *Corale #4, preludio for cello and string orchestra*) to express various musical constraints [1].

We believe that this preliminary work shows the interest of a framework enabling the systematic building and processing of abstract spaces that appear in musical analysis. 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. As a matter of facts, one of the benefits of the spatial approach is the expressiveness and the concision of the constraint formulation. Only one rule is used to build the underlying space and only one rule is enough to specify the path to search.

We are currently working on spatial approach for the enumeration of Hamiltonian paths in various chord networks. We are also investigating the detailed geometry of the clusters of AIS. Future works include the validation of the approach on more examples and on larger scale problems, as well as the development of additional spatial constructions that may be necessary to handle further musical formalizations.

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

References

1. Albini, G., Antonini, S.: Hamiltonian cycles in the topological dual of the tonnetz. In: Chew, E., Childs, A., Chuan, C.H. (eds.) Mathematics and Computation in Music, Communications in Computer and Information Science, vol. 38, pp. 1–10. Springer Berlin Heidelberg (2009)
2. Bigo, L., Spicher, A., Michel, O.: Spatial programming for music representation and analysis. In: Spatial Computing Workshop 2010. Budapest (Sep 2010)
3. Callender, C., Quinn, I., Tymoczko, D.: Generalized voice-leading spaces. Science 320(5874), 346 (2008)
4. De Hon, A., Giavitto, J.L., Gruau, F. (eds.): Computing Media and Languages for Space-Oriented Computation. No. 06361 in Dagsthul Seminar Proceedings, Dagsthul, `http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=2006361` (3-8 september 2006)
5. Gent, I.P., Walsh, T., Hnich, I., Miguel, I.: CSPLib: a problem library for constraints. web page at `http://www.csplib.org` (accessed in january 2011)
6. Giavitto, J.L.: Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In: Rewriting Technics and Applications (RTA'03). LNCS, vol. LNCS 2706, pp. 208 – 233. Springer, Valencia (Jun 2003)
7. Giavitto, J.L., Michel, O.: MGS: a rule-based programming language for complex objects and collections. In: van den Brand, M., Verma, R. (eds.) Electronic Notes in Theoretical Computer Science. vol. 59. Elsevier Science Publishers (2001)
8. Giavitto, J.L., Spicher, A.: Systems Self-Assembly: multidisciplinary snapshots, chap. Simulation of self-assembly processes using abstract reduction systems, pp. 199–223. Elsevier (2008), doi:10.1016/S1571-0831(07)00009-3
9. Jedrzejewski, F.: Mathematical Theory of Music. Delatour (2006)
10. Mazzola, G., et al.: The topos of music: geometric logic of concepts, theory, and performance. Birkhäuser (2002)
11. Mazzola, G.: La vérité du beau dans la musique. Delatour (2007)
12. Morris, R., Starr, D.: The structure of all-interval series. Journal of Music Theory 18(2), pp. 364–389 (1974), `http://www.jstor.org/stable/843642`
13. Munkres, J.: Elements of Algebraic Topology. Addison-Wesley (1984)
14. Otterström, T.: A theory of Modulation. Da Capo press, Inc. (1935)
15. Riotte, A., Mesnage, M.: Formalismes et modèles musicaux. Delatour (2006)
16. Schiff, D.: The Music of Elliott Carter. Faber and Faber (1983)
17. SCW: 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` (accessed in january 2011)
18. Spicher, A.: Spatial computation and classification of all-interval series. Tech. rep., LACL, Univ. of Paris-Est (Jan 2011)
19. Spicher, A., Michel, O., Giavitto, J.L.: Declarative mesh subdivision using topological rewriting in mgs. In: Int. Conf. on Graph Transformations (ICGT) 2010. LNCS, vol. 6372, pp. 298–313 (Sep 2010)
20. Truchet, C., Codognet, P.: Musical constraint satisfaction problems solved with adaptive search. Soft Computing - A Fusion of Foundations, Methodologies and Applications 8, 633–640 (2004), `http://dx.doi.org/10.1007/s00500-004-0389-0`
21. Tucker, A.: An abstract approach to manifolds. Annals of Mathematics 34(2), 191–243 (1933)
22. Tymoczko, D.: The geometry of musical chords. Science 313(5783), 72 (2006)