AN ASSOCIATIVE APPROACH TO COMPUTER-ASSISTED MUSIC COMPOSITION

Kevin Dahan
Université de Paris VIII
CICM

ABSTRACT

This article presents a new conceptual approach to formal computer-assisted music composition. Procedural and object-oriented methods for computer music composition are recalled, then compared to associative modelling. A strategy of implementation for this particular approach is then described, using the concept of lexemes, and an example of implementation is described. Finally, strategies for efficient graphical representation are evoked.

1. INTRODUCTION

This paper addresses the issue of how to represent in a simple but efficient way the complexity of the computer music composition task. Whereas most of the tools available nowadays use a *directional* approach to composition conceptualization, we argue that a new approach is needed, based on an association-based mechanism.

Freeing the composer from a restrictive and hard-to-restate perspective, the associative approach not only permits him to focus on forging relationships between objects rather than morphologically change them, but equally to work outside an otherwise rigid time framework.

Most of the techniques developed nowadays focus on hard real-time approach, and hence permit sound manipulation at almost no cost in terms of computing power. While composition is usually thought as an "off-time" process, there is little software providing support for non-realtime manipulation, and providing the user with a non-constrained, unoriented interface. Using an "associative", instead of a directional approach to "off-time" composition, where the directionality is not provided by an external framework, but directly contained in objects and their associations, new ways of envisaging the composition process emerge.

We will briefly introduce the associative approach as an extension of the well-known object-oriented approach, and explain the basis of an associative system for composition. We will then describe how a software tool should be designed to fully support this paradigm, with implications on the internal mechanisms, and on the graphical interface.

2. DIRECTIONAL VS. ASSOCIATIVE APPROACH

The fundamental difference between the directional and associative approach is the weight given in the latter to *relationships* between objects, rather than on the morphology of the objects themselves, pervasive in the former.

2.1. Objects

Objects are the basis of most modern computing approaches, object-oriented programming [6], and more recently aspect-oriented [4], and have been proved very successful in the computer music field (software such as *MAX/MSP* or *OpenMusic* are examples of these techniques), where complex situations quickly arise.

As [5] recaps, in object-oriented systems:

- The object support encapsulation,
- The object is self-contained,
- The focus is on structure instead of function,
- The focus is on methods instead of processes.

These features, applied to computer-assisted composition environment [1], freed the composer from a number of constraints that were bound to earlier algorithmic approaches [1]. For example, a "simple" transformation applied on a particular sound had to be descriptively expressed each time the transformation was to take place, when using the procedural (i.e. algorithmic-based) method [2], whereas the object-orientation permitted a much more dynamical definition of the events to occur, encapsulating the description of the transformation in the object itself and providing its description with a contextual reference informing on "when" to apply the transformation.

As a result, the object-oriented paradigm leads to a "directional" approach to composition, as networks formed by objects obey the "flowchart" paradigm [7], and this can be described as a deterministic succession of events [3].

[1] from now on referred as *CACE*.

[2] Music-V style languages are emblematic of this very deterministic conception.

[3] The use of entropy or random generator in this case doesn't invalidate the comment, as they only change the end-result of the composition, and do not change the overall procedure of the composition.

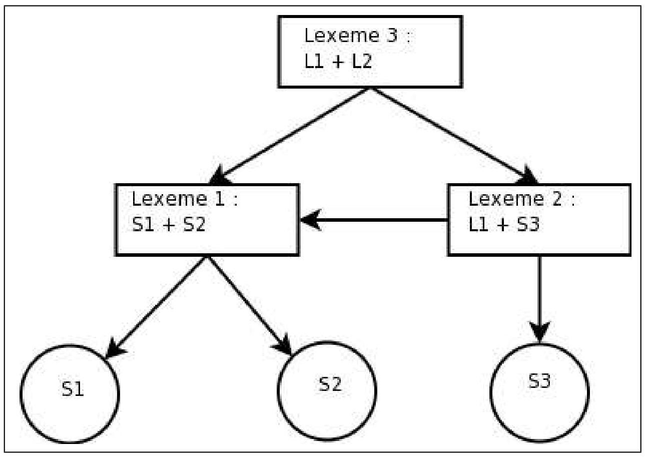

Figure 1. Example of lexeme structuration

2.2. Lexemes

What we call *lexeme* is a building block constituted of two objects - called *morphemes*, linked by a defined process.

2.2.1. Description

While the description of this compound may seem an integration of a procedural element in an otherwise purely object-oriented system is in fact greatly different, as the link may be described as object, and, as such, may be instantiated. Considered as objects, links can therefore be dynamically changed, leading to the lexeme being considered itself as an morphemic entity, and consequently be linked. A basic representation of possible resulting associations with three initial morphemes is shown figure 1.

Typical links may include (but not limited to):

- Time relations,
- Filtering processes,
- Conditional connectors,
- Interactive inputs.

Due to its object nature, links may be dynamically updated, as objects can.

2.2.2. Rationalization

The dynamic nature of link can thus be expressed as a special case of object, where the link object B is expressed as a bidirectional object, when acting as a link:

$$B \sim \overleftrightarrow{B} \tag{1}$$

Thus, lexemes can be expressed logically as the compound of the objects A, B, C:

$$\Lambda_{A,B,C} = \langle A^{\overleftrightarrow{B}} C\rangle\,^{4} \tag{2}$$

[4] And can be simplified to:
$$\Lambda_{A,B,C} = \langle ABC\rangle$$

A common situation is to link an object to an existing lexeme. If Λ_1 is defined as $\Lambda_1 = \Lambda_{A,B,C}$ and is linked by D to an object E, we have:

$$\Lambda_{\Lambda_1,D,E} = \langle \Lambda_1{}^{\overleftrightarrow{D}} E\rangle = \langle\langle A^{\overleftrightarrow{B}} C\rangle^{\overleftrightarrow{D}} E\rangle \tag{3}$$

If the same link (here B) is used when defining such a lexeme, we obtain:

$$\Lambda_{\Lambda_1,B,E} = \langle \Lambda_1{}^{\overleftrightarrow{B}} E\rangle = \langle\langle A^{\overleftrightarrow{B}} C\rangle^{\overleftrightarrow{B}} E\rangle \tag{4}$$

This introduces the notion of *magnitude*. A lexeme is said to be of magnitude *n* if the same link is used *n* times to define intricate lexemes. We use the following notation to designate this special case, with L as the link object:

$$\Lambda^{n}_{A,L\ldots,X} = \langle\langle A,\ldots,X\rangle \cdot n\overleftrightarrow{L}\rangle\,^{5} \tag{5}$$

Complexity arises in the situation where lexemes are defined by means of *lexemes of lexemes*. Let's examine the situation where a lexeme $\Lambda_1 = \Lambda_{A,B,C}$ is linked to a lexeme $\Lambda_2 = \Lambda_{E,F,G}$ by using the link D:

$$\Lambda_{\Lambda_1,G,\Lambda_2} = \langle \Lambda_1{}^{\overleftrightarrow{G}} \Lambda_2\rangle = \langle\langle A^{\overleftrightarrow{B}} C\rangle^{\overleftrightarrow{D}} \langle E^{\overleftrightarrow{F}} G\rangle\rangle \tag{6}$$

Ultimately, a complete composition would be thought as a *meta-lexeme*, constituted by the compound of all the "local" lexemes, arranged in time (using a time link t), such as:

$$\Lambda_{\Sigma} \supset \langle A,\ldots,Z\rangle = \langle\langle A,\ldots,X\rangle \cdot n\,\overleftrightarrow{t}\,\rangle = \Lambda^{n}_{A,t\ldots,Z} \tag{7}$$

The associative approach to composition, is, in a certain way, an extension of the current *transformational* approach to sound structuration. In that case, the composer works at different time scales using the same structuration mechanisms, and the associative approach permits to work effectively in the same manner at any abstraction level. Despite the rationalization we exposed here, its use is almost intuitive, as it is very natural to define sound structures as associations of smaller elements, and thinking transformations on sound structures the same way[8].

3. APPLICATIONS

We used the concept of lexemes implicitly in *The Sketcher* prototype [2] and [3] [6] . At the time, the system lacked the conceptual ground now found in the lexemic approach to be effective. However, a certain number of mechanisms and graphical choices certainly helped the emergence of the associative approach to *CACE*.

[5] *n* is defined to be the number of items linked, since there may be multiple instances of the same objects, it is needed.

[6] An early version of the prototype is available as a Linux LiveCD at: http://www.dcs.shef.ac.uk/ guy/mistres/sketcherv0.99.iso

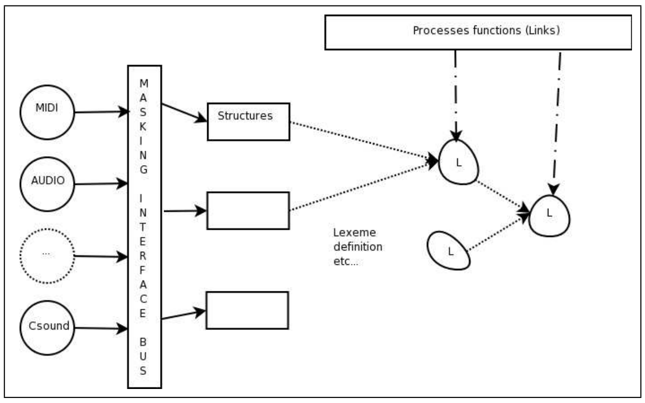

Figure 2. Internals of a lexeme-based system

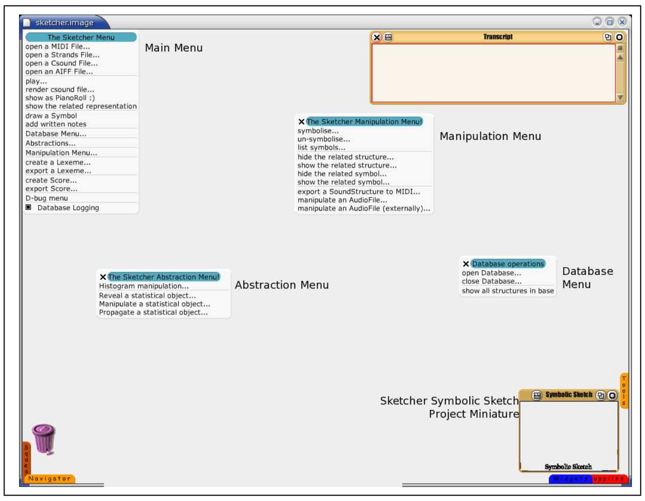

Figure 3. The sketcher main window

3.1. Internal Mechanisms

The main problem when dealing with the lexemic approach is that of heterogeneous data types, of common use within the computer music field. Most composers work with pools of different data structures and types (MIDI files, Audio files, sound synthesis descriptive language source *la Csound*, and so on...), and it is needed to be able to work indifferently of the data structure to be addressed in order to implement correctly the lexemic superset.

To this end, an abstract class is defined, acting as a masking interface for low-level calls to I/O mechanisms, and providing some sort of "system bus" for higher-level structures to call indifferently any data type needed. Several solutions exist for calling audio streams, by providing an intermediate representation consistent with CMN, MIDI streams, or "note list" abstraction (typically, Fourier-based methods gives good results). Spectrum information are not taken into account for the representational system only, and are still used and pertinent in all manipulations - this vital information for composition is not at all lost, and the user can specify to which extent he wants the representational "middle-ground" to be specific. A representational sketch of such a system is given in figure 2.

3.2. Graphical Aspects

In order to coherently represent the complexity of the lexeme-based approach (and in a non-invasive method), a redefinition of graphical canons used in common computer music application is in order.

Instead of constraining the user into a semantically rich workspace, cluttered with iconic representations, a blank workspace is presented, without any indication. The *Sketcher* interface, though relying on menus for specific action (exportation, link definition, etc...), is entirely based on this approach [7] , as shown figure 3.

A specificity of *The Sketcher* is to let the user freely associate various data types - which are not meaningful to the system - such as pictures or text and to define associations between them and meaningful objects. It allows the composer to be able to switch from a high level of representation (i.e. the score level, in which he defines notation convention), to the low-level representation in which he deals with structure definition, sound manipulation and lexeme creation. At all times, the "score" representation is available in the right hand corner of the screen, working as a reminder of the overall form the user wishes to achieve.

Such a radical redefinition of GUI for *CACE* is needed to coherently represent the multiple levels of representation and interaction taking part in a composition environment based on object interactions. Horacio Vaggione [9] indirectly expressed the need for more advanced means of control over complex sound manipulations and association in an object-based *CACE*, by pointing the limits of the current technological choices, loosely based on old 1st and 2nd order cybernetics assumptions. Hence, the patch-cord (analogical modelling) or track-based representations are in some way invalidated by Vaggione's ontological remarks.

In order to permit the composer to fully exploit the associative approach, there is absolutely no constraint in the arrangement of structures: placement on the desktop is insignificant, as there are no frequency nor time scales. Order is defined solely by the use of lexemes, and not visually represented if the user choose not to.

4. CONCLUSION

The lexemic (or associative) approach to computer music composition, though based on a simple paradigm of structure linking, has strong implications at all levels of the *CACE* design: structuration and articulation of events in micro- or macro-time scales can be defined by the same relations, with no morphological-induced constraints; there is no need for intermediary representations.

Firstly, it creates a new paradigm for the composer: instead of dealing with the composition as a *flowchart*

[7] A similar, though less "unguided" graphical interfaces is found in the *Hyperscore* software, http://www.media.mit.edu/hyperins/ts-hyperscore/.

diagram, it permits *interactive structures* to be used instead. Directionality is not construed by relying on external mechanisms (i.e. track-based editing, score-following processes), but is created by the interactions (i.e. the lexemes) that are developed between the different objects.

As a side-effect, complete redefinition of the current strategies used in the graphical interfaces for composition software is needed. Since meaning and directionality is mostly contained in objects and lexemes, there is little interest in providing an interface cluttered with directional information (i.e. tracks, frequency or time scales), unless provided as an *additional* functionality.

A new software is currently in development[8], at first experimenting the associative approach provided by the use of lexemes, then enhancing the current representational strategy, and finding new representation methods. Investigation on the possible use of lexemes in the context of electroacoustic music analysis is also underway.

5. ACKNOWLEDGEMENTS

The author wishes to thanks Adrian Moore for reports and encouragements on *the Sketcher* software, and the *MISTRES* group at Sheffield University for numerous inputs.

6. REFERENCES

[1] Budon, Osvaldo. “Composing with Objects, Networks, and Time Scales: an Interview with Horacio Vaggione”, in *Computer Music Journal*, Vol. 24/3, Cambridge, USA, 2000.

[2] Dahan, Kevin, Brown, Guy J., and Eaglestone, Barry. “New Strategies for Computer-Assisted Composition Software: A Perspective”, in *Proceedings of the International Computer Music Conference*, Singapore, 2003.

[3] Dahan, Kevin, Eaglestone, Barry, Brown, Guy J., Moore, Adrian J., and Ford, Nigel. “Design for a Computer Music Composition Tools System”, *MOSART EU Report d36, T6 2nd deliverable*, MOSART EU-RTN Project, 2003.

[4] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, J.-M., and Irwin, J. “Aspect-Oriented Programming”, in *Proceedings of the European Conference on Object-Oriented Programming*, Jyväskylä, Finland, 1997.

[5] Kristensen, Bent Bruun. “Associative Modeling and Programming”, *Proceedings of the 8th International Conference on Object-Oriented Information Systems (OOIS’2002)*, Montpellier, France, 2002.

[8] Using the Ruby programming language.

[6] Meyer, Bertrand. *Object Oriented Software Construction*, Prentice-Hall, Upper Saddle River, USA, 1997.

[7] Puckette, Miller. “Using Pd as a score language”, in *Proceedings of the International Computer Music Conference*, Götebord, 2002.

[8] Truax, Barry. “Time and Electroacoustic Music”, *Academy of Electroacoustic Music*, Bourges, 1999. See: http://www.sfu.ca/ truax/bourges3.html.

[9] Vaggione, Horacio. “Some Ontological Remarks about Music Composition Processes”, in *Computer Music Journal*, Vol. 25/1, Cambridge, USA, 2001.