On the Need for Visual Formalisms in Music Processing

Article *in* Leonardo · April 1999

DOI: 10.1162/002409499553109

2 authors:

Mira Balaban
Ben-Gurion University of the Negev
71 PUBLICATIONS **430** CITATIONS
SEE PROFILE

Michael Elhadad
Ben-Gurion University of the Negev
96 PUBLICATIONS **2,425** CITATIONS
SEE PROFILE

On the Need for Visual Formalisms for Music Processing*

Mira Balaban Michael Elhadad
Dept. of Mathematics and Computer Science
Ben-Gurion University of the Negev
P.O.B. 653, Beer-Sheva 84105, Israel
mira@cs.bgu.ac.il elhadad@cs.bgu.ac.il
(972)-7-461622, 461627, fax: 972-7-472909

Technical Report FC-94-14

Abstract

The design of Computer Music Environments (CMEs) for supporting interactive composition remains a challenge, due to the complex nature of music ontology, and the needs to support creative processes initiated by naive users, all in real time, using complex engineering technology. Elsewhere we argued that CMEs should be provided with a well defined methodology, that relies on techniques for modeling, knowledge representation, and implementation, in artificial intelligence and software engineering.

In this paper we concentrate on the essential role of visualization in CMEs, and the need for a well defined visualization methodology. We claim that visualization deserves a specialized theory, that is based on music ontology, and is independent from the concrete, implemented graphical interface. We support our claim through a critical evaluation of the visualization aspect in the MAX and DMIX CMEs, and by providing examples demonstrating the richness, flexibility, and power of our approach.

*This research was supported, in part, by the Israeli Ministry of Science and Arts, and by the Paul Ivanir Center for Robotics and Production Management at Ben-Gurion University of the Negev.

1 Music Ontology – A Multi-Dimensional Paradigm

Over the last decade, research and applications in computer music have flourished. Today's computer music tools support a large variety of applications, from tutoring and instruction, through automatic accompaniment, to algorithmic and interactive composition. Research in computer music includes models for music processing, that are based on a variety of approaches in connectionism, artificial intelligence, and software engineering.

Computer-music initiated new paradigms of research and composition. A major contribution of computer-music to cognitive musicology has been the focus on *music activities* [8] – in contrast to traditional musicology, which concentrates on the study of existing pieces. That is, the new approaches to cognitive musicology deal with the dynamics of music making, rather than with the static final product. Interactive composition is a particularly attractive composition paradigm, because it enables direct contact between the composer and the composition, and gives the composer capabilities for real time control or performance of the composition. It is also a platform for investigating the composition process. Existing tools for interactive composition make extensive usage in state of the art graphical and audio capabilities, provided by the computer industry.

At the current stage, the design of Computer Music Environments (CMEs) for supporting interactive composition remains a challenge. The problems result from the complex nature of the music ontology that is being processed, and the needs to support creative processes initiated by naive users, all in real time, using complex engineering technology.

In [3] it was argued that the design of CMEs requires the development of a well defined methodology, that relies on techniques for modeling, knowledge representation, and implementation, in artificial intelligence and software engineering. In this paper we concentrate on the essential role of visualization in CMEs, and the need for a well defined visualization methodology. We believe that visualization is central to the desired interactive nature of CMEs, and as such, deserves a specialized theory. This theory must be based on the status of visualization in music ontology, and be independent from the concrete, implemented graphical interface.

1.1 The Problematics of the Design of User Interface to CMEs

Music ontology is a complex phenomenon that can be approached from multiple viewpoints, including acoustics, mathematical analysis of sounds, music parameters, human conception of music, music expectations, music ideals, music activities, music analogies, visual analogies, social values, emotions, cognition, music aesthetics, music conventions, and many more. Clearly, a CME cannot account for the full phenomenon, but certain dimensions such as sound level, time, hierarchy, tex-

tual descriptions, music activities, and visualization of music, seem unavoidable. A major challenge for *interactive composition* is the need for simultaneous manipulation of this variety of dimensions.

A second challenge for CME design lies in the targeted audience. The *end-user problem* is especially demanding in such environments, since there is an uncompromised need for an artistically appealing human interface.

The multidimensionality issue combined with the user access problem define the challenge of CME design: In a single interface, one must be able to encode and manipulate information about temporal and hierarchical structure, sound parameters at different abstraction levels (from low-level MIDI parameters, to high-level reference to timbre and pre-defined instruments), performance instructions (that can refer to opaque emotional states) in an easily accessible and tailorable way.

These different levels cannot be described independently of each other, since they ultimately describe the same ontological object. Instead, they must be viewed as different perspectives on a common partially known object (cf the *partiality principle* discussed in [3]). At the user-interface level, one must be prepared to integrate and coordinate different media to describe the relation between these different levels. Thus, for example, structured graphics can be used to encode the temporal and hierarchical structure of a musical piece, user-defined icons can be used to describe pre-defined instruments, analogic graphic (X-Y charts) can be used to describe low-level sound parameters, text can be used to encode higher-level emotions.

The user interface research programme for CME resides in the development of proper technology to describe the available *representation devices*, the way to *compose* such building blocks into readable and easily manipulated structures, and the way to *coordinate* the different media into a coherent presentation.

1.2 Music Visualization and Music Activities

Visualization and *activity* are essential for supporting the dynamic aspect of the composition process. To refine our understanding of the role of visualization in a CME, the notion of *affordance* is particularly relevant. Affordances are "properties of the environment that offer actions to appropriate organisms" [5].

Specifically, for computer supported composition, we are interested in studying the composition activities which are the building blocks of the composition process. Very often, such composition activities cannot be described in a textual manner. These activities include, besides general actions, such as cut and paste, repetitions, modifications, and extracting patterns or templates, complex action that are "dimension specific", such as application of a parameter or a music function, merge of parameters, and merge of pieces. For many of these activities, visualization becomes a critical

enabling medium that defines an otherwise unrecognized activity by showing its affordance. The *slapping* activity introduced in DMIX and discussed below offers a striking illustration of this point: slapping is an easy to grasp visual operation but it is hard to explain analytically.

1.3 A Methodology to Tackle the Complexity of CME Design

Experience gathered in the fields of Artificial Intelligence (AI) and Software Engineering (SE) in the development of complex systems, shows that a theory of engineering such systems should be developed. There is a general agreement that at least three levels of information should be distinguished:

1. *Ontology* – the portion of the real world music phenomenon being accounted for. The ontology should clarify what are the music entities, distinctions, operations, relations, and constructs that the system deals with. The ontology is always partial, and can be constantly augmented.

2. *Representation* – formalisms that describe the ontology. They should admit an incremental, open nature. In particular, they should provide a proper account for the visual, the audio, and the operations dimensions.

3. *Implementation* – realization of the representation level, supporting internal levels, and an end-user audio-graphical-gestural-textual level. The implementation directly interacts with the sound level dimension in the ontology.

We claim that a theory of engineering CMEs, should provide a proper account for the visualization and activities dimensions at the representation level, and not leave them to implementation level. The separate level of *visual representation* emerges from ontological conventions, and serves as the affordance for intuitive and real-time music activities.

In the rest of the paper, we review the way visualization was introduced in existing CMEs, MAX and DMIX (Section 2). In both cases, visualization was handled at the level of implementation and is not supported by an appropriate representation formalism. We then explain why such a distinct visual formalism is necessary and demonstrate its advantages for CMEs (Section 3).

2 Music Visualization in MAX and DMIX

MAX and DMIX are two distinguished CMEs that support different aspects of interactive composition. Visualizations of music and activities are central to both. Below, we shortly describe each system, and emphasize their contributions and limitations. The main point we make in this

section is that while these systems demonstrate impressive visual capabilities, they are restricted by the initial visual metaphors of their designers. Due to the complication of the ontology, they cannot be consistently extended - hardly by programmers and certainly not by end-users. End-user extensibility of the visual interface to CMEs is, however, necessary, due to the artistic needs of the targeted audience and we claim that it requires an overall theory of visualization in CMEs.

2.1 MAX

MAX ([17, 10, 18]) is a procedural graphical programming environment for developing real-time music systems. The underlying metaphor for MAX graphics and control is taken from analog synthesizers. MAX graphics consists of patches, which are collections of boxes, connected by lines. MAX computation is based on message passing along the lines that connect boxes. That is, the patches graphics corresponds to processes that fire each other along the lines connecting their boxes. The real-time character is enabled by letting box objects directly access MIDI I/O streams, or DSP devices. Figures 1, 2 and 3 are examples of MAX patches.

1. **Figure 1**: In this patch every MIDI input signal, that consists of the three items of *pitch*, *velocity*, and *channel*, fires the *notein* object, from which the three items flow to the *noteout* object. The pitch line undergoes a modification, that is determined, interactively, by the *slider* attached to the pitch. The user plays, interactively, with the slider (using mouse operations), and the amount added to the pitch is determined by the slider level. The MIDI output signal is comprised from the input velocity and channel, and the slider determined pitch.. The function performed by that figure is:

$$(p,\ v,\ c) \leftarrow MIDIinput(pitch,\ velocity,\ channel);$$
$$MIDIoutput(slider(p),\ v,\ c);$$

2. **Figure 2**: This patch describes a repetition of a note, input by a MIDI signal. The repetition bound is determined interactively by the *counter* slider, and the delay between repetitions is determined interactively by the *delay* slider. The main function of the patch (ignoring the counter and delay sliders) is:

$$\begin{array}{ll} (p,\ v,\ c) & \leftarrow MIDIinput(pitch,\ velocity,\ channel); \\ repeat & until\ counter < 0 \\ & MIDIoutput(p,\ v,\ c),\ delay,\ decrement(counter); \end{array}$$

composition as a whole. Visualization of elements and complex actions is an essential byproduct. Like MAX, DMIX is not equipped with a theory of operation and abstraction.

DMIX and MAX demonstrate opposite approaches to the design and role of computer-music tools - MAX as a low-level uni-dimensional tool, DMIX as a broad multi-dimensional system. In both of them, however, the lack of underlying theory turns possible generalizations into ad-hoc extensions. For example, the abstraction operation is theoretically well understood. It requires clear syntax for defining the syntactic unit on which the abstraction is performed, and formal definition of the resulting abstraction and its meaning. Had the visual components of MAX or DMIX been well defined, the operation of *visual abstraction* could easily be added[2]. Likewise, it is impossible to provide different visualizations than those proposed by the system. In MAX, you can see the procedural description encoded in the network of boxes and connections, but you cannot derive from it a description of the types of sounds that can be produced or a time-line of the music. In DMIX, the history of operations that created the piece is lost - so a sequence of interactive manipulations cannot be turned into an activity.

We claim that the intrinsic complex demands of engineering CMEs, require a well-defined methodology of separation and coordination among the different levels of information. The visual shortcomings of DMIX and MAX seem to result from confusion of levels of representation and processing. In particular, the development of visual end-user interfaces should be based on careful distinctions between levels of knowledge that concern visualization. There are levels that capture music visualization of various dimensions. These levels should be distinguished from a graphical level that concern the precise layout of objects, or the precise geometric shapes used to form icons for music actions. In the next section we describe the fundamental methodology of visual formalisms, and show how it can support multiple visualizations, and provide affordance of visual operations.

3 A Methodology for Visual Formalisms

In contrast to *ad-hoc* non-explicit approaches to visualization, we propose to develop a *visual language* based on the KR level of music structures. The methodology to define a visual language is inspired by work in graphical presentation, such as [9] and [20], where a graphic presentation is composed automatically from graphical devices selected according to the semantics of database relations and user intent, and work in natural language processing, where language is viewed as a

[2]MAX reminds early languages like APL, that provided attractive means for expressing small computations, but proved incapable of supporting large software engineering tools.

device to convey the speaker's communicative intentions (cf. e.g., [1], [4]). Our approach is also inspired by Harel's work on hygraphs [6].

The visual language methodology maps a well-defined set of semantic relations to a selected set of topological relations. This well-defined mapping allows bidirectional translation between a KR structure and a visual diagram. Only certain topological relations are given meaning in this mapping - which form the set of graphical devices available in the visual language. For example, hierarchical relations in the KR level can be expressed as connected nodes in a graph-like visual language (yielding the classical nodes and lines representation of hierarchies), if the visual language contains the topological relation of *connectedness*. If, in contrast, the visual language uses the topological relation of *containment*, the same hierarchy in the KR level will be represented by embedded boxes, that can possibly overlap. Note that in these two examples, the exact location of each object in the visual representation are not completely determined by the underlying semantic relations in the KR.

For this reason, one must distinguish between the graphic level and the topological structure of a diagram. A separate algorithm, called the *layout* component is responsible to compute the exact geometric composition of the visual diagram under the topological constraints specified by the visual language. Typically, the layout algorithm attempts to satisfy aesthetic and stylistic criteria. Thus, if the visual language only gives significance to connectedness, the left-right location of nodes on the diagram cannot be interpreted in any significant way, and it is left to the layout algorithm to compute the exact coordinates of each element in the diagram.

In music, traditionally, certain topological relations must be respected: left-right ordering is always interpreted as temporal precedence, top-bottom alignment as temporal simultaneity. So our visual language must respect these conventions. But new graphical devices can be introduced to express additional dimensions of the music descriptions - these include color, three-D rendering, containment of pieces. These graphical devices must be interpreted with respect to the KR level. Without such a well-defined representation level, the visual representation will remain fuzzy. In particular, there can be no clear distinction between significant topological relations and layout artifacts[3].

The overall structure of the method is shown in Figure 6, which shows the three distinct levels of processing defined. In addition, the figure illustrates the role of visualization as an affordance for music activities. The critical step is that visual representations allow the user to define *abstractions*,

[3]Indeed, in the above mentioned users discussion of MAX, users complain on confusion in understanding the impact of a patch layout on the resulting music. This is due to the lack of separation between visual and graphical levels in MAX

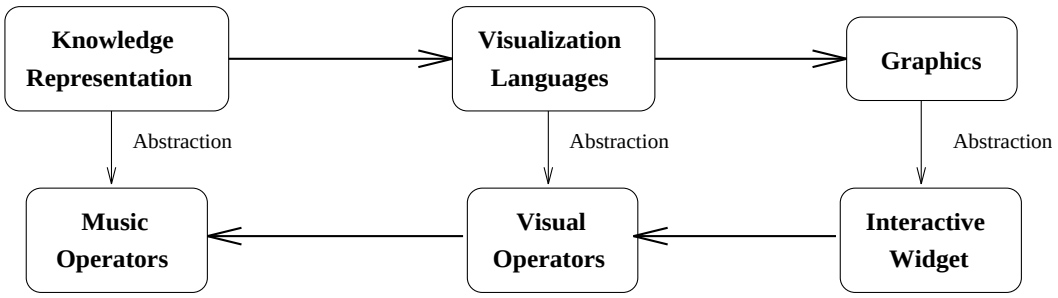

Figure 6: Overall Architecture of the Proposed Visualization Methodology

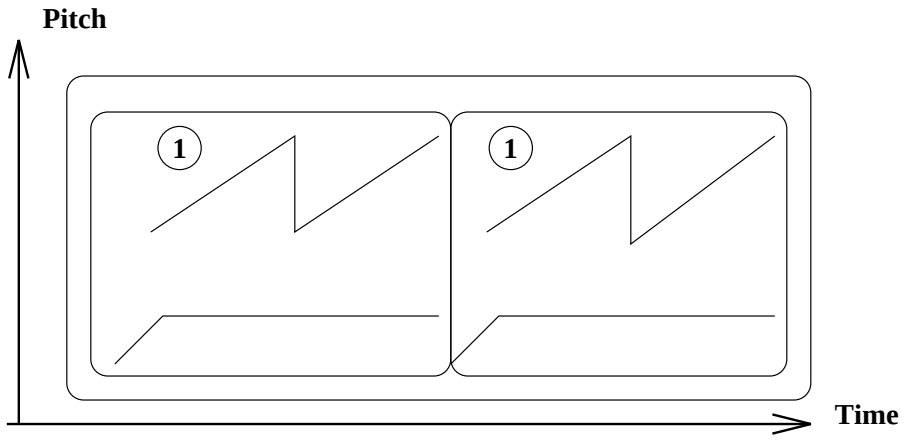

Figure 7: A Time-Pitch Visualization of the first bar of Bach's First Prelude. Structure is Shown as Containment

which can be turned into operators. At the visual level, a user uses abstraction to turn a constant diagram into a visual operator which can then be applied to other visual elements. Because the visual level is derived from the knowledge representation in a sound manner, the visual operator thus defined corresponds immediately to a formal operator at the knowledge representation level. The visual level thus empowers the user with the possibility to define new functions at the knowledge representation level without working on any formal language.

To illustrate the interrelation between abstraction and visualization, we now present an example based on Bach first Prelude. Note that we are not attempting to present a musicologist interpretation of the finished work, but rather to imagine how a composer could use visual tools to construct a piece by defining new music operators and applying them to some raw material.

First, Figure 7 shows one visualization of the first bar of the Prelude along the dimensions of Time and Pitch. The same visual language shows a simple form of hierarchical structure in the form of embedded boxes. In this visual language, the circled label 1 indicates that the same piece is repeated twice. This visual language is probably the closest to the traditional music notation, shown in Figure 8.

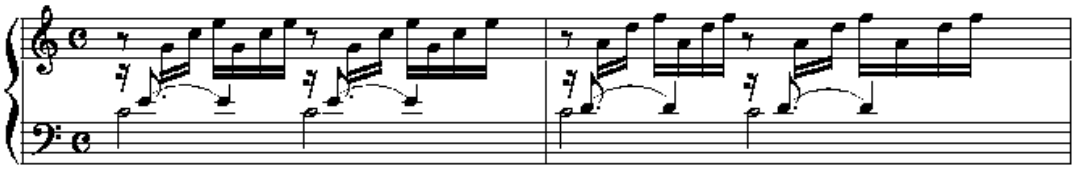

Figure 8: Traditional Music Notation for the Opening of Bach's First Prelude

Since the Prelude is notable for its intricate structure, we also show an alternative visualization of the chord structure of a single bar, using a different visual language, in Figure 9. Still with the same visual language, we show a hierarchical view of a single bar, in Figure 10. Both of these figures are based on the formalism of Structured Music Pieces introduced in [2]. In Figure 9, temporal operators are shown in circles. The horizontal bar at the top denotes the operation of temporal sequence. Each operator is connected to a number of parameters. In this case, the repetition of the same piece twice in a row is marked by the two arrows leaving the sequence operator and reaching the same chord node. A textual rendering of the complete structured piece is as follows, going from bottom-up: Start from five notes (s1 to s5) and join them into a note-list. Turn this note-list into a chord of duration 1/2. Repeat the same chord twice in a row. Similarly, in Figure 10 the operator vertical bar denotes simultaneous temporal concatenation, the operators 1st and 2nd extract the first and second note from a chord, and the operator subchord extracts a range from a chord (the range is specified in the second parameter, in the figure 3-top).

Each operator in these descriptions corresponds to a primitive music activity. Abstraction at the visual level is realized by circling a subset of the overall network and turning it into a new operator. Thus, in Figure 9, the user can turn the whole network into a *Bach_Bar* operator which accepts as input five notes and a duration, and produces as value a music piece of the form

$$\textbf{seq}(\ \textbf{chord}(\textbf{list}(s1, s2, s3, s4, s5), dur),\ \textbf{chord}(\textbf{list}(s1, s2, s3, s4, s5), dur)\ ).$$

The new operator can be written down formally as a function:

$$\begin{aligned} Bach\_Bar = \quad & \textbf{function}(s1, s2, s3, s4, s5, dur) \\ & \quad \textbf{let } ch = \textbf{chord}(\textbf{list}(s1, s2, s3, s4, s5), dur) \\ & \quad \textbf{in seq}(ch, ch) \\ & \textbf{end} \end{aligned}$$

Once new operators are defined through abstraction, they are given a name and can be used in exactly the same way primitive operators can be. In the Bach Prelude example, all bars except

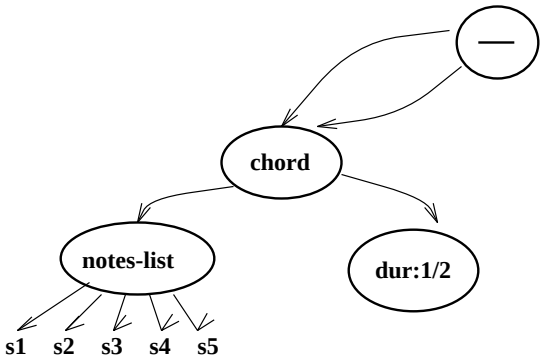

Figure 9: Chord structure of a single bar in Bach first Prelude

for the last three can be produced by applications of the *Bach_Bar* operator. The first bar is: $Bach\_Bar(C1, E1, G1, C2, E2, 1/2)$, and the second one is $Bach\_Bar(C1, D1, A1, D2, F2, 1/2)$ (the numbers denote octaves). Note that in our example, the operators deal with the structural aspect of the pieces being constructed, because we applied the abstraction operation to a visualization of the structural dimension. A similar abstraction can be performed over other dimensions as well. Structural operators can then be used in other views, leading to a cross-fertilization of the dimensions. So for example, in Figure 7, the two dimensions of pitch and structure are shown in the same view. It is possible to think of the embedded views as the application of the *Bach-bar* structural operator designed in Figure 9 on the time-pitch view. In this cross-dimension application, it is possible to decide what notes will be given as input to the *Bach-bar* operator by embedding a portion of the time-pitch diagram.

4 Conclusion

In this paper we considered two notable CMEs for interactive composition, that emphasize the essential role of visualization in interactive composition in general, and for the *affordance* of complex operations, in particular. While these two systems show the potential of visualization in supporting interactive composition, they remain limited since they do not allow for true generalizations that involve new dimensions, new visual metaphors, or introduction of complex operations.

We claim that due to the complex, multidimensional, partial nature of the music ontology, CMEs must be incremental, open-ended systems. This goal can be achieved only by supplying CMEs with an underlying theory, that respects the separation of the ontology and representation from the implementation. The visual dimension requires an additional level of representation, that is separate from the concrete graphical implementation. We demonstrate the richness and flexibility provided by this approach, by using different graphical expressions as an account for

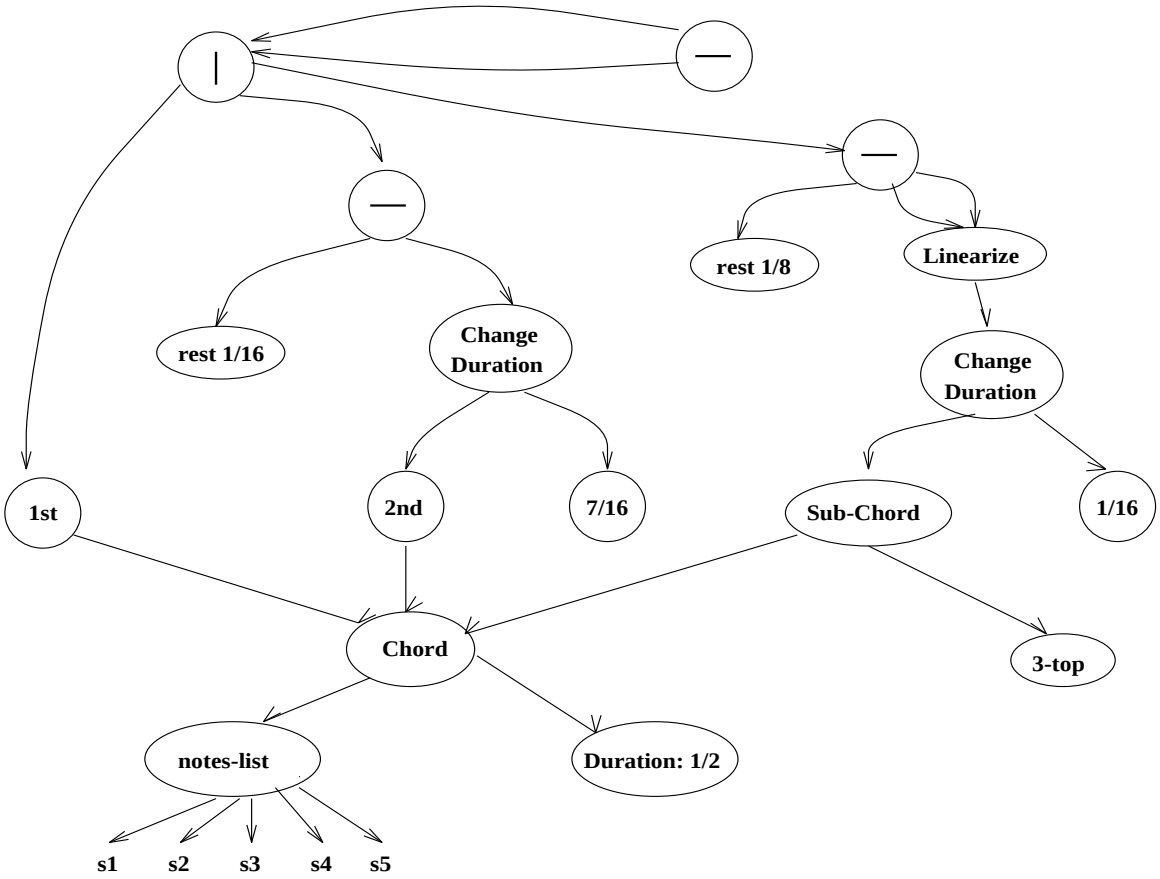

Figure 10: Hierarchical view of a single bar in Bach first Prelude

hierarchy (containment, in Figure 7, and connectedness, in Figures 10, 9). We demonstrate the power of this approach by showing how visual operators can be defined through the operation of visual abstraction.

The overall message of this paper is that visualization is a hard and essential task of CMEs. It must be provided with a syntactic and semantic theory, that ties it to the rest of the internal KR level, on one side, and to the graphical user interface, on the other side.

References

[1] J.F. Allen and C.R. Perrault. Analysing intentions in utterances. *Artificial Intelligence*, 15(1):143–178, 1980.

[2] M. Balaban. Integrating terminologies into f(rames)-logic knowledge bases. Technical Report in preparation, Department of Mathematics and Computer Science, Ben-Gurion University, Beer Sheva, Israel, 1994.

[3] M. Balaban. Introducing formal processing into music – the music structures approach. *Computer Music Journal*, 1995. accepted.

[4] P. Cohen and C. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[5] J.J. Gibson. *The ecological approach to visual perception.* Houghton Mifflin, New York, 1979.

[6] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–529, 1988.

[7] D.J. Kurlander. *Graphical Histories.* PhD thesis, Columbia University, Computer Science, 1992.

[8] O. Laske. Artificial intelligence and music: A cornerstone of cognitive musicology. In M. Balaban, K. Ebcioglu, and O. Laske, editors, *Understanding Music with AI: Perspectives on Music Cognition*, pages 3–28. MIT–AAAI Press, 1992.

[9] Jock Mackinlay. *Automatic Design of Graphical Presentations.* PhD thesis, Stanford University, 1986.

[10] Opcode, Inc., Palo Alto, CA. *MAX Documentation.*

[11] D.V. Oppenheim. Dmix: An environment for composition. In *International Computer Music Conference*, pages 1–8, Columbus, Ohio, 1989. San Francisco: International Computer Music Association.

[12] D.V. Oppenheim. Quill: An interpreter for creating music-objects within the dmix environment. In *International Computer Music Conference*, pages 1–6, Glasgow, Scotland, 1990. San Francisco: International Computer Music Association.

[13] D.V. Oppenheim. Shadow: An object-oriented performance system for the dmix environment. In *International Computer Music Conference*, pages 1–6, Montreal, Canada, 1991. San Francisco: International Computer Music Association.

[14] D.V. Oppenheim. Compositional tools for adding expression to music. In *International Computer Music Conference*, San-Jose, CA, 1992. San Francisco: International Computer Music Association.

[15] D.V. Oppenheim. Dmix: A multi faceted environment for composition and performing computer music: Its design, philosophy, and implementation. In *Computer Music Days*, pages 1–10, Delphi, Greece, 1992.

[16] Y. Orlarey, D. Fober, S. Letz, and M. Bilton. Lambda calculus and music calculi. In *International Computer Music Conference*. San Francisco: International Computer Music Association, 1994.

[17] M. Puckette. The patcher. In *International Computer Music Conference*, pages 420 – 425. San Francisco: International Computer Music Association, 1988.

[18] M. Puckette. Combining event and signal processing in the max graphical programming environment. *Computer Music Journal*, 15(3):68–77, 1991.

[19] R. Rowe and B. Garton. Guest editorial note: The mins of max. *Computer Music Journal*, 17(2), 1993.

[20] D. Seligmann. *Automatic Generation of Intent-Based 3D Illustrations*. PhD thesis, Columbia University, 1993.