Towards A Framework for Handling Musical Expression

Daniel V. Oppenheim, James Wright
Computer Music Center, IBM T. J. Watson Research Center
Route 134, Yorktown Heights, NY 10598
music@watson.ibm.com, jwright@watson.ibm.com

Abstract: It is hard to imbue computer music with expressive gestures. We are proposing a framework specifically to support that need. This framework is based on three components: a content representation called Nuance, a framework for rendering expressive elements called LeNNY, and various user interfaces for specifying and manipulating expressive gestures.

Keywords: music, expression, composition, representation, interactive

1. The Score and Musical Expression

Our musical experience of instrumental music is stimulated by an interplay of at least two factors: the score as notated by the composer, and its expressive interpretation during a performance. It is well understood that scores are highly amenable to formal musical analysis. Performance practice, on the other hand, cannot be reduced into a formal framework and is traditionally passed orally from teacher to student. Composers of instrumental music deal with the formal aspects of musical composition but always while considering how they will actually sound when rendered by a performer.

In computer music the composer often produces a complete work that is rendered onto tape and will never be subject to further interpretation. In such cases the composer must not only compose his work but also perform it. A system for composing such music must include facilities for what Oppenheim termed *composed expression (*Oppenheim 92). Most existing systems for computer music rely heavily on the score model. This model does not readily lend itself to the specification of musical expression (Desain & Honing 1991, Honing 1992). This limitation does not stem from the score model itself, but rather from the composer's inability to figure out the 'correct' parameters needed to produce a desired expressive gesture.

Our work deals specifically with *composed expression*: it is a framework for specifying and editing musical expression. This is work in progress, rather than an all inclusive solution to the problem of musical expression. We present some initial design ideas, and expect many changes will occur as the work progresses and our understanding improves.

2. Our Approach

We approach the problem of musical expression on three different levels:

1. The **Nuance** content representation, with specific extensions for musical expression.
2. The **LeNNY** framework is built on top of Nuance specifically for rendering Nuance scores onto various output devices (i.e. synthesizers, signal processors). The output from LeNNY contains expressive details, much like a live performance.
3. Several kinds of **user interfaces** for specifying, visualizing, and editing musical expression. Different editors can access multiple levels of a compositional structure.

Our approach differs from traditional solutions in the following ways:

1. We distinguish between Score Values, Expressive Attributes, and Effective Values. A *Score Value* is the traditional P-field parameter (e.g. pitch = 440Hz). *Expressive Attributes* modify Score Values (*legato, crescendo, brighter,* etc.). The *Effective Value* is the Score Value after it has been transformed throughout the system to include all the Expressive Attributes. The Effective Value is analogous to the result of a live performer playing a Score Value.
2. The modeling of expression is cumulative and uses a hierarchical structure. Often, the Expressive Attributes are not wholly contained in individual notes, but may reside in a hierarchy of layers, each of which can add its own interpretation or meaning. For example, a Phrase can include Expressive Attributes that will affect each of its notes, and yet also be contained within a Section that contains additional Expressive Attributes that further modify the expres-

sive detail. In this case, playing the same Phrase when disconnected from the Section will produce different Effective Values than when played as part of the Section.

3. Each kind of expressive attribute is interpreted using a *Context* that specifies how that attribute may be evaluated, manipulated and rendered. *Contexts* are also cumulative and can be nested anywhere within the compositional hierarchy.
4. The end musical result, i.e. the Effective Values, is achieved through the interaction of three separate entities: the *Score Values*, *Expressive Attributes*, and *Music Contexts*.

3. The Nuance Music Representation

Nuance is a simple, open-ended representation based on a few fundamental constructs, that supports the evolution of families of dialects specialized for different needs and domains. It is being developed to support on-going work in the creation, performance, manipulation and analysis of music and other time-based media.

Some primary objectives of Nuance include:

- Extending the "note event" model to include what happens inside a note, between successive notes, and how these are affected by the overall musical context.
- Explicit modeling of expressive attributes.
- The use of simple building blocks and inspectable, role-based specifications to promote clarity, generality and extensibility.

Attributes, Bundles and Contexts are the three major constructs of Nuance (fig. 1). A simple example would use Attributes to represent parameters like pitch or onset, Bundles to represent musical structures from notes to complete works, and Contexts to provide the frames of reference within which these elements are interpreted, manipulated and performed.

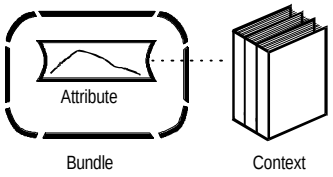

Figure 1

An individual "note" may be modeled as a single Bundle of Attributes, with each Attribute capturing a given kind of musical detail (fig. 2).

An **Attribute** defines a functional role within a content domain, such as pitch, onset or loudness. An Attribute is represented using one or more kinds of **Domain Value**. For example, CentsPitch, MidiPitch, DiatonicPitch and HertzPitch are all potential Domain Values of a Pitch

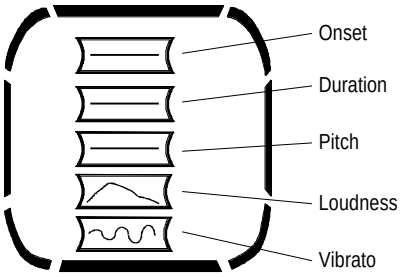

Figure 2

attribute. Domain Values related to a common attribute type are said to be **compatible** and may be freely inter-converted. DomainValues are somewhat like the MusicMagnitudes used in DMIX (Oppenheim 1992, 1996) and SmOKe (Pope 1996); a key difference is the use of explicit Contexts in Nuance.

Attribute values can vary across the extent (duration) of a single bundle, producing a time-variant attribute **contour**. Attribute contours support dynamic inflections such as envelopes and cyclic modulations.

A **Bundle** is a container with temporal extent. It may contain Attributes and/or other Bundles. Bundles may be nested to construct higher-order entities such as chords, motivic fragments, phrases and sections (fig. 3). Bundles may also be used to represent expression contours, harmonic analyses, or other kinds of performance, analytical or compositional information. A given bundle can be contained by multiple other bundles, so long as the resulting bundle network is a directed acyclic graph (DAG).

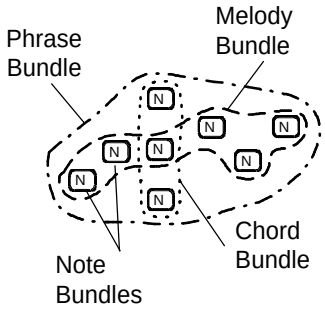

Figure 3

Bundles also provide a scoping mechanism. An attribute associated with a bundle affects all of its nested bundles.

Contexts define the environment within which Attributes are evaluated and manipulated. A given Context holds the metadata (semantic information) and support mechanisms for a

particular Attribute, and for the Domain Values which embody that Attribute.

A Context includes:

- Role-based specifications defining the runtime structure of related Domain Values.
- Methods for value conversion between compatible (role-related) Domain Values.
- Combining rules for evaluating nested attributes and attribute contours.
- Lookup tables and other referential data needed to support value conversion and manipulation.

For example, the Pitch context might define one or more scales (interval patterns), a Midi pitch table, spellings of accidentals, Middle C tuning and intonation table. Given this information, conversions between MidiPitch, HertzPitch, CentsPitch and DiatonicPitch are trivial.

Contexts may be nested as well, so that different portions of a work can override or modify global environmental information when appropriate.

4. Expression Modeling In Nuance

Expressive qualities within a single note may be modeled directly using a single attribute with time-varying extent. This includes gestures such as pitch envelopes and cyclic modulations. Expressive contours involving a series of note elements are modeled using c*ompound attributes*, which result when a nested bundle and one or more containing bundles have attributes of compatible kinds. Each such attribute is then considered to be an *aspect* of a single compound attribute.

The base (innermost) aspect represents the scored value of the attribute, while the other aspects represent expressive contours or deviations. The *effective* value of the compound attribute is determined by evaluating all compatible in-scope attributes, using combining rules in the relevant Context.

For example, a work might use a global tempo map, a "groove template" defining a specific rhythmic micro-structure, and three distinct section–, phrase– and bar– level contours (fig. 4). Certain attributes model the flow of time at a given structural level, while others model expressive deviations from that flow. Taken together, they form six aspects of a single compound Onset attribute. The effective onset for each note is determined by evaluating the four higher-level attributes along with the nominal ('scored') onset of that particular note and aligning the result with a location in the rhythmic microstructure grid.

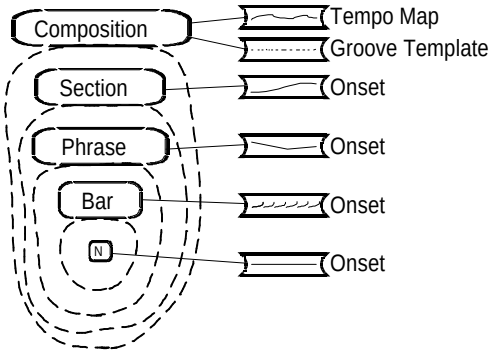

Figure 4

5. The LeNNY Rendering Architecture

The Nuance music representation provides the basic building blocks for dealing with musical expression. The LeNNY architecture is a framework built above Nuance specifically to utilize this potential. LeNNY renders Nuance scores onto specific synthesis and DSP devices.

One may view a Nuance score as a 'conventional' score augmented with an additional structure is attached that specifies expressive details. This second structure includes the Expressive Modifications and references to Context objects, both of which are scattered throughout the compositional hierarchy.

A Performer is a LeNNY object that parses the Nuance structure, evaluates and reconciles all the Expressive Modifications and relevant Contexts, and finally produces the actual synthesis parameters (fig. 5). Similar ideas can be found in (Dyer 1991, Anderson 1993). We intend to use one performer for each synthesis instrument type, so that different performers may interpret given Expressive Attributes in different ways (a tuba performer may interpret a *legato* attribute differently than a violin or piccolo performer, and a piano performer will ignore *vibrato* attributes).

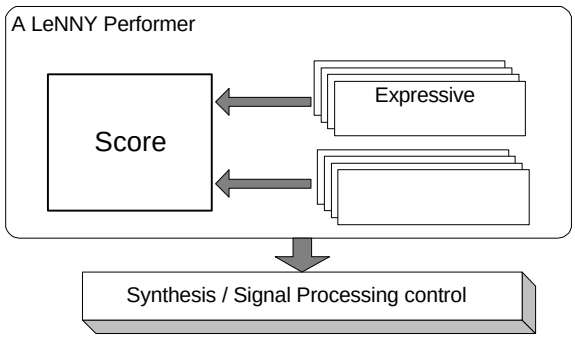

Figure 5

6. The User Interface

The user interface is a key element for allowing a composer to actually obtain 'musical' results. The user interface must provide access not only to note events and their parameters, but also to all of the expressive attributes and music contexts that are spread throughout the hierarchy of the composition. We expect that many commonly used tools will have to be adapted so that they can handle expression (e.g. piano roll, note list and common music notation editors). In addition we expect that new types of editors will be needed (e.g. graphical editors for direct manipulation of attribute contours.)

7. An Imaginary Expression Editor

Figure 6 is an imaginary example of one possible user interface using a DMIX Structure-View. This view presents a Bach composition (bottom black rectangle) comprised of two inner sections: opening and middle section (see black rectangles in diagram center). The opening section contains two parts: a melody and a bass (light gray rectangle in top left corner); the middle section contains 4 subsections, each with melody and bass (top right).

The composer listens to the music and decides to phrase the opening section. To so this, he or she applies a performance-by-rule BlockClosure [1, numbers refer to Figure 6]. However, the need for further changes becomes clear after hearing the result. A second layer of Expression is added which combines with the first and explicitly sets the onset times of several selected events [2]. Next, the *loudness* in the opening and middle sections is refined using a DMIX Modifier [3], which is edited interactively to obtain the desired result. It happens that the change in loudness has had some unwanted side effects, so the composer decides to add yet another layer of Expression to 'correct' the articulation [4]. This is done with a real-time Expression Modifier: the composer adjusts the articulation using a MIDI controller while listening to the entire music playing. In [5] a BlockClosure is applied. This algorithm may take into account musical attributes in other sections of the work (or anywhere in the DMIX environment) to determine some desired attributes of the inner section within the middle section. And finally another Modifier is used to determine the *tempo* [6] of the middle section.

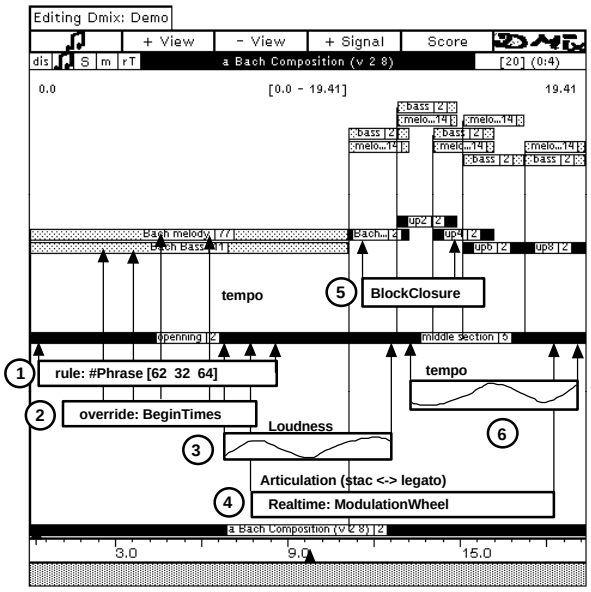

Figure 6

8. References

Anderson, T. (1993) "E-SCAPE: An Extendible Sonic Composition And Performance Environment." Ph.D. thesis, University of York, UK.

Desain, P., & Honing, H. (1991) "Towards a calculus for expressive timing in music." Computers in Music Research, 3, 43-120.

Dyer, L. (1991) "An Object-Oriented Real-time Simulation of Music Performance Using Interactive Control", Ph.D. thesis, CCRMA, Stanford University, Report STAN-M-78.

Honing, H. (1992). "Expresso, a strong and small editor for expression", Proceedings of the ICMC, San Jose, CA.

Oppenheim D. (1992), "Compositional Tools for adding Expression to Music in DMIX." Proceedings of the ICMC, San Jose, CA.

Oppenheim, D. (1996) "DMIX-A Multi Faceted Environment for Composing and Performing Computer Music." Computers and Mathematics with Applications, 32(1):117-135.

Pope, S. (1996) "Object-oriented music representation." Organised Sound, 1(1):55-68, Cambridge University Press, UK.