On continuous musical control of discrete musical objects

Peter Desain & Henrik Horning

NICTI Computational Linguistics, Faculty of Arts
Nijmegen University University of Amsterdam
P.O. Box 9104 Spuibestraat 134
NL-6500 HE Nijmegen NL - 1012 VB Amsterdam

This paper focuses on the gap between discrete, symbolic representations of music (like common music notation) and continuous, numerical representations (like audio and control signals). The distinction is also found in computer music composition systems with discrete event-oriented systems, like Max, at one end and continuous signal-oriented systems, like Csound, at the other. The often ignored realm in between consists of the control of the levels of modulation and articulation: how time-varying parameters are linked to specific parts of the discrete structure, and how they behave under transformation of this structure (for example, how a particular kind of phrasing of a rhythmic fragment depends on the duration of it). These levels are still little explored (in computer composition as well as in perceptual modeling), although the main problems are well-identified and some solutions have been proposed (e.g., Rodet & Coote, 1984; Dannenberg, 1986; Scarlatti, 1972). We think that the difficulty of using this kind of control is due to the lack of formalisms to express different kinds of musical knowledge at these levels.

Generalized time functions In Desain & Horning (1992) we proposed the idea of control functions of multiple times. These so-called generalized time functions (GTF) are functions of start-time and duration of the object they are linked to and the actual time. The need for this kind of formalism is illustrated by the "vibrato problem"; a continuous control function used to describe a vibrato should add more periods when it is stretched or linked to a longer musical object, while, in contrast, a glissando control function expressed by the same means should be elastically stretched. When time-varying functions are also made dependent of start-time and duration, these different behaviors are made explicit. GTF's can be plotted as three-dimensional surfaces when the absolute start-time is ignored. They describe a control value for every point in time given a certain time interval. A specific surface describes the behavior under time transformation (like stretching the discrete object it is linked to). In Figure 1 this surface is shown for a simple sineoidal vibrato, a sinusoidal glissando, and a more complex function (the transition function in Figure 5).

Figure 1: Control values (bottom to top) as a function of duration (front to back) and time elapsed since start (left to right): a) a vibrato, b) a glissando, and c) a complex control function.
Integration with discrete musical objects

We replaced the ad hoc flat event-list representation, used to illustrate the workings of the GTF in Desai & Honig (1992), by a set of discrete hierarchical musical objects. Together these musical objects and GTFs can form alternating layers of discrete and continuous information. For example, a phrase can be associated with a continuous amplitude function, while consisting of notes associated with their own envelope function, which are in turn divided into small sections each with its specific amplitude behavior. The lowest layer could even be extended all the way down to sound samples.

Both musical objects and GTFs have standard ways of being combined into new ones. Basic musical objects like note (with parameters for duration, pitch, amplitude and other arbitrary attributes depending on the synthesis method used) and pause (a rest) can be combined into compound objects using the time structuring constructs (sequential) and (parallel). Furthermore, new musical objects can be added with their own specific behavior under transformation (see, for instance, trill in Figure 2b). The basic GTF's, like a linear ramp, can be combined into more complex control functions using a set of combinators (like compose, concatenate, multiply, add, etc.) or by supplying GTF's as arguments to other GTF's. The actual decision of describing a certain musical aspect in a continuous or discrete way is sometimes arbitrary. A trill, for example, can be described as one note with an alternating control function for its pitch (see trill example in Figure 2a). Or it can be described as a discrete musical object consisting of several notes filling the duration of the trill (see trill example in Figure 2b). Both descriptions will correctly add more elements (periods or notes) when stretched.

Figure 2. Trill example.

Figure 3. Parametrization (a) and transformation (b) of continuous attributes of basic musical objects.

Passing control functions to attributes of musical objects: parametrization vs. transformation

There are alternative ways of passing information of control functions to musical objects. One method is to pass a GTF directly as an attribute to, for example, a note (see Figure 3a). A disadvantage of this method when building complex musical objects is that parameters that are controlled from the outside need to be known in advance - the encapsulation of the musical object needs to be parametrized.

An alternative method is to generate musical objects with simple default values and to obtain the desired result by transformation. For instance, in Figure 3b, an amplitude...
transformation on a simple note with constant amplitude yields a note with a specific amplitude contour.

A disadvantage of this method is that the transformation is in principle applied to the whole musical object and can not differentiate its behavior with regard to specific aspects of that object. Thus both methods need to be available.

![Graphical output of note examples a and b.](image)

Figure 4. Transformation (a) and parametrization (b) of continuous attributes of compound musical objects.

Passing information from musical objects to control functions

When using the parametrization method, a GTF is passed the start-time and the duration of the note that it is linked to as attribute. For the transformation method, a GTF used in a transformation is passed the start-time and the duration of the compound object it is applied to. Thus, to give it the same power as the transformation method, one needs a construct to span a control function around a whole musical object, and apply it as parameter to the different parts. Figure 4a shows the natural use of the transformation style in applying an amplitude envelope to a compound musical object, Figure 4b shows the construct with attached-time-funs that enables linking of a control function to an atomic musical object before it is referred to as parameter in parts of that object.

Passing control functions laterally between musical objects: the transition problem

Once control functions are incorporated into musical objects as attributes, it is sometimes needed to allow reference to them from within other musical objects. The construct with attribute-time-funs allows this, taking care that transformations on musical objects do affect later reference to their attributes properly. Control functions are in principle defined for the interval of the object they are linked to. But when they are referred to from musical objects at other time positions it is useful to be able to evaluate them at other times. This will support extrapolation, for instance needed in describing, what happens in a transition between notes (cf. Rodet & Coine, 1984). In Figure 5 a simple example of such a transition transformation is shown. It applies to two notes, derives the pitch control function from both, and calculates an interpolation between them (using the cross-fade GTF combinator). A comparable example is a portamento, in which one musical object (a note) needs to have access to an attribute (pitch) of another musical object (the next note).

One has to note that it is easy to write the example above using a top-down communication, defining the control functions in advance before they are referred to in several places after that. However, this communication has to be foreseen and extrapolation of musical objects cannot be maintained.

In Figure 5 first two untransformed notes are shown, followed by a rest and the same notes with a transition.

Passing control functions upstream: the compressor problem

Sometimes a global transformation of an attribute needs to refer to the attribute values of the complex musical object that it is applied to. This bottom-up communication is needed, for instance, when one wants to compress loudness of musical object. This is an
amplitude transformation that needs access to the amplitude of the object itself. The total amplitude at any time depends on the sum of the amplitudes of parallel parts. So the amplitude transformation of a specific note relies on its parallel context. In Figure 6 a possible definition of a compressor transformation is given showing make use of this type of communication. The overall amplitude is calculated by summing amplitude contours for parallel sub-structures and concatenating them for sequential ones. Then the inverse amplitude is applied to the musical object. First the original musical object is shown, followed by the same object being compressed.

A comparable example is the intonation problem, where a similar type of communication is needed to describe how parallel voices adjust their intonation dynamically (Ashley, 1992). Note that this type of transformation can only be realized when musical objects are returned as data structures (and not when they produce output as side-effect).

Reference


Postscript
In a number of computer music systems some of the representational problems described above have been solved either explicitly or implicitly. A project that might contribute to the understanding of these systems is the construction of a compact music composition systems questionnaire as a list of prototypical problems and questions. One of the ideas behind this work document is to gain more understanding of existing systems, recognize the problems and state them. It already benefited from input from Rolf Danneberg, and the music representation group at IRCAM (Célestin Angyan, Gerhard Eckel and others) and can be obtained via honing@max-lat.uzh.ch.

---

(defun musical-object ()
  (p (note :amplitude .1 :pitch 0)
   (note :duration .75 :amplitude .1 :pitch 59)
   (note :amplitude (ramp 0 .2) :pitch 62))
  (note :amplitude (ramp 0 .8) :duration .5 :pitch 58)))

(defun compress (musical-object)
  (with-attribute :time-funs (((ramp :amplitude \*'concatenator \*'sumner)
                             musical-object))
    (amplitude musical-object (time-fun: '1 amp))))

Graphical output of: (s (musical-object) (pause) (compress (musical-object)))

Figure 6. Compressor example.