The Hierarchical Functional Inheritance Model

A Mathematical Abstraction for Computational Process Music

Carl McTague[*]

May 20, 2000

Abstract

By combining the notion of event indexing with a hierarchical model with arbitrary inheritance functions, we present a simple, yet general and powerful solution to the event definition problem in music, facilitating pervasive application of process to musical composition.

A Proposition

We begin with a simple proposition.

Proposition 1. *For all pieces of music (more precisely a single observer's perception of a performance of any such piece[1]) there exists a unique continuous real-valued function $f \in C(\mathbb{R})$ mapping a time-dependent variable t to the position $f(t)$ of a fixed point on a sound-sensitive membrane.*

How do we define such a function? We view the creative process of musical composition as the act of defining such a function and concern ourselves with this task for the remainder of this paper.

An overwhelming body of music is composed of *notes*; the music may be decomposed into a finite number of discrete, atomic objects with, among other properties, an associated timing interval (i.e. a starting time and duration). This is the inspiration for our definition of an *event.*

[*]Copyright © 1999-2000 by Carl McTague (carl@mctague.org)

[1]For simplicity we assume single-channel audio, although, for most humans, a musical experience is perceived in two channels (one per ear).

Definition 1. *An* event $e \in C(\mathbb{R})$ *is a real-valued continuous function composed by the association of a function e^* with an interval I denoted $e = (e^*, I)$ defined*

$$e(t) = \begin{cases} e^*(t) & \text{for } t \in I \\ 0 & \text{for } t \notin I \end{cases}$$

with $e^*(t) = 0$ for t on the boundary of I. We denote the set of all events $\mathbb{E}$.

We can then define the function f simply as the sum of a collection of events $E \subset \mathbb{E}$,

$$f = \sum_{e \in E} e$$

With this definition of f we can see the task of musical composition as the construction of a set of events. For the remainder of this paper, we concern ourselves with the construction of finite collections of events.

Event Indexing

Because $\mathbb{E}$ is a *very* large set, we find it useful to index a subset of it with a more familiar set.

Definition 2. *Let S be a set. Then we call a map $p : S \to \mathbb{E}$ an* event indexing function *and we call each $a \in S$ an* event representative *of its image $p(a)$. Further, we denote by $\mathbb{P}_S$ the set of all event indexing functions of S.*

With these definitions we reduce the task of musical composition to selecting a familiar set S, creating at least one event indexing function $p \in \mathbb{P}_S$, and constructing a subset T of S. For we can then define f with

$$f = \sum_{a \in T} p(a).$$

Note that this approach is very common. Consider composition in the Cmix environment. When a composer creates a Cmix instrument, he creates families of functions (each Cmix instrument typically represents an entire family of functions; specific functions tend to be selected through arguments.) Then in his Cmix score file he defines timing intervals and associates them with functions by linking them to Cmix instruments with particular arguments. Indeed, even the act of composing at the piano may be seen in this way.

Hierarchy and Inheritance

We have now set the stage for our primary object: the presentation of a powerful and efficient technique for constructing finite subset of event indices and pairing them with event indexing functions: a hierarchical structure.

Imagine a tree structure whose leaves represent events constructed in such a way that each event is determined by its position within the tree. We accomplish this by associating to each branch in the tree an *inheritance function* and to each leaf an event indexing function. We then propagate an event representative through the tree from the root. The representative is altered and redistributed at each branch by the corresponding inheritance function. When the propagating representatives reach leaves of the tree, they're made into events by the corresponding event indexing functions. We make this formal.

Note 1. *For the remainder of this section, let n be a positive integer and let S be a set.*

Definition 3. *An* inheritance function *is a function*

$$R_n : S \to S^n$$

which to each element of S assigns an n-tuple of elements of S.

Definition 4. *An* event hierarchy ψ_n *is an association of an* inheritance function R_n *with an n-tuple of objects $(c_1, \ldots, c_n)$ written*

$$\psi_n = (R_n, (c_1, \ldots, c_n))$$

where each $c_1, \ldots, c_n$ is either itself an event hierarchy or an event indexing function. We denote the set of all event hierarchies $\mathbb{Y}_n$.

We now make formal how to *flatten* event hierarchies into finite collections of events.

Definition 5. *Let $a \in S$ and let ψ_n be an event hierarchy with $\psi_n = (R_n, (c_1, \ldots, c_n))$. Choose $b_1, \ldots, b_n \in S$ such that*

$$R_n(a) = (b_1, \ldots, b_n).$$

Then we define the function $\Xi_a : \mathbb{Y}_n \to P(\mathbb{E})$, which to each event hierarchy associates a collection of events by

$$\Xi_a(\psi_n) = \begin{cases} \bigcup_{k=1}^{n} \Xi_{b_k}(c_k) & \text{for } \psi_n \in \mathbb{Y}_n \\ \{\psi_n(a)\} & \text{for } \psi_n \in \mathbb{P}_S \end{cases}$$

We say that Ξ flattens *ψ_n.*

An Example

We now provide a simple example. Let $S = \mathbb{R}^2$. We define an event indexing function $\sigma \in \mathbb{P}_S$ with

$$\sigma(x, y) = (E^*, [x, x + y])$$

where E^* is a waveform that sounds like a firm human clap in a magnificent concert hall. That is, $\sigma(x, y)$ is an event representing a clap during the interval of time $[x, x + y]$.

Next we define two inheritance functions, α_n and β_n. Let $(x, y) \in S$. Then let

$$\begin{aligned} \alpha_n(x, y) = \Big(\Big(x, \frac{y}{n}\Big), \Big(x + \frac{y}{n}, \frac{y}{n}\Big), \ldots \\ \ldots, \Big(x + (n-1)\frac{y}{n}, \frac{y}{n}\Big) \Big) \\ \beta_n(x, y) = ((x, y), \ldots, (x, y)). \end{aligned}$$

Finally, for convenience, we define a shorthand notation.

$$\begin{aligned}\alpha(c_1,\ldots,c_n) &\rightarrow (\alpha_n,(c_1,\ldots,c_n))\\ \beta(c_1,\ldots,c_n) &\rightarrow (\beta_n,(c_1,\ldots,c_n))\end{aligned}$$

We have now developed an incredibly versatile tool for rhythm synthesis. As a simple example, consider the event hierarchy

$$\psi = \alpha(\alpha(\sigma,\alpha(\sigma,\sigma)),\beta(\alpha(\sigma,\sigma),\alpha(\sigma,\sigma,\sigma))).$$

We can flatten ψ to obtain a set of events T by computing $T = \Xi_{(0,24)}(\psi)$,[2] which yields

$$\begin{aligned}T =&\{\sigma(0,6),\sigma(6,3),\sigma(9,3),\sigma(12,6),\\ &\sigma(18,6),\sigma(12,4),\sigma(16,4),\sigma(20,4)\}\end{aligned}$$

Finally, we define f for this rather short, pedantic piece by summing the elements of T. That is,

$$f = \sigma(0,6) + \cdots + \sigma(20,4).$$

Conclusion

In conclusion, we have developed a remarkably versatile abstraction for algorithmic musical composition. We first identified the central problem in musical composition as the definition of a single function. Next, we formalized the notion of musical events and introduced the event indexing functions. We then developed event hierarchies with inheritance functions. Finally, we illustrated a single powerful application of these tools for rhythm synthesis.

Acknowledgments

I would like to acknowledge the inspiration of two composers, Philip Glass and Steve Reich. Their works have revolutionized how I hear and write music. I would particularly like to acknowledge Philip Glass's score to Godfrey Reggio's film *Koyaanisqatsi* and Steve Reich's *Music for Mallet Instruments, Voices and Organ* which I have heard hundreds of times and know by heart.

[2]We chose the tuple $(0,24)$ simply to avoid fractions (for typographical cleanliness)!