Temporal type constructors for computer music programming

Eli Brandt

June 2002

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Roger Dannenberg, Chair
Guy Blelloch
Robert Harper
Perry Cook, Princeton University

Abstract

Existing high-level ways of doing computer music programming have sacrificed generality, making some algorithms impossible to express. My idea is that they fail due to type systems that are inadequate. I propose a way that is high-level without being restrictive: applicative programming with temporal type constructors.

Temporal type constructors are type constructors that introduce temporal structure. I propose a set of three—event, vector, and infinite vector—and give examples of the variety of musically useful types that they form. Temporal type constructors let the programmer describe exactly the data type that is key to a given algorithm, and manipulate it from a viewpoint outside of time.

I describe a programming system called Chronic that provides these temporal type constructors. Code examples written in Chronic show how some computer music algorithms can be expressed unprecedentedly well: several times shorter than their equivalent in a low-level language, and with a smaller semantic gap between the program and the idea—and these are algorithms that cannot be expressed at all in the high-level languages.

I then go into some detail about how Chronic is implemented, and why an infinite vector is not just a lazy list. I arrived at an infinite-vector implementation that does dynamic dataflow with a self-modifying graph of O'Caml objects, but the decisions along the way may be of interest to implementors facing different trade-offs. Interesting points include how to do block computation, efficient fan-out to multiple consumers, and in-place update. Vector comprehensions are necessary future work; they would give an order-of-magnitude speedup in some common cases. I explain why lazy vectors are a necessary step towards real-time operation, and why real-time remains a hard problem in Chronic's programming model, particularly due to its standpoint outside of time.

The novelty here is using type constructors to introduce temporal structure into the data, so that pure applicative programming becomes possible over the entire domain of computer music. This kind of programming is especially well suited to problems involving data with significant structure or multiple levels of abstraction. It has expressiveness from the construction of complex time-structured types suited to the problem at hand, and generality because every type can be deconstructed down to the individual audio sample. Many interesting computer music programs become easier to write and to read.