The Representation of Rhythmic Structure

David Wessel, Vijay Iyer, Jeff Bilmes, and Matt Wright

Center for New Music and Audio Technologies (CNMAT)
University of California, Berkeley
www.cnmat.berkeley.edu

Overview

A data structure and associated processes are presented for the representation of rhythm. This representation involves a temporal unit or cell containing a specified number of isochronous pulses. These pulses or temporal atoms, *tatums* , form the indices for each vector in a set of rhythmic property vectors associated with each cell. The rhythmic property vectors presented here include: a probability vector containing the probabilities that events will occur on given *tatums* ; an accent vector containing the intensities of the events; a duration vector containing times between onset and release of each event; a deviation vector containing temporal offsets from the *tatum* grid; and additional property vectors more directly related the sound material. Continuous functions control tempo variation. We show that this representation can be applied to a large variety of music, that it provides perceptually potent control, particularly with respect to metrical structure and the temporal fine structure associated with "feel". With proper constraints the representation can be used as an analysi-synthesis model.

A MAX implementation will be presented and demonstrated with special emphasis on the organization and manipulation of the data in the representation.

Motivation

Traditional music theory, traditional music notation, and most modern computer-based music sequencers fail to address a number of the perceptually important features of rhythmic organization. The main shortcomings involve the lack of an explicit representation for temporal fine structure. Traditional musical practices, be they of the western tradition or of the diverse muscal traditions throughout the world, leave the composition of temporal fine structure to performance practices, practices that are transmitted from musician to musician by example in the form of an "oral" tradition. And, the now ubiquitous music sequencers typically fail to provide much in the way of additional specification over traditional notation and when they do they either involve the fine hand tuning of note placement and event durations or the crude application of random deviations.

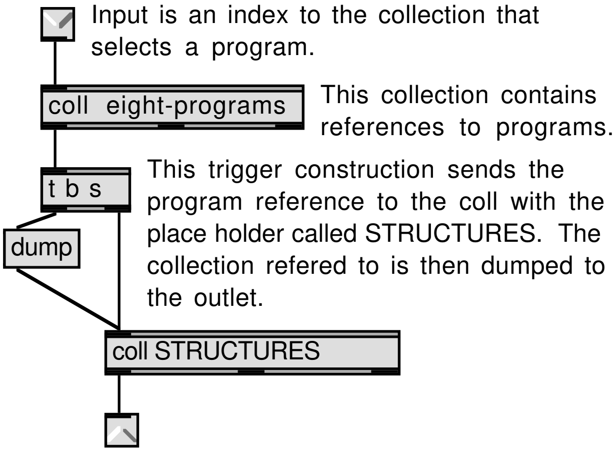
Our goal is to provide a more complete representation of rhythmic structure and to supply process oriented models that can be used in live-performance computer music contexts where improvisation and reactive musical dialogues are central concerns. The ICMC97 paper by Iyer *et al* provides additional information on the model and its motivations. In this talk I will demonstrate the approach with a number of examples and describe a technique for organizing data in MAX that played a central roll in programming the MAX implementation of the model.

Organizing Data in MAX

The MAX programming environment for the Macintosh has been frustrating for many in their attempts to manipulate large amounts of structured data. Certainly for those like myself who came to MAX after LISP, working with a rich data-base seemed at first to be a daunting task. In this section, I would like to present some techniques that may not be well known to the MAX programming community that offer an easily implemented way to organize data and use it efficiently in a real-time performance context.

The technique uses the **coll** object developed by David Zicarelli at the time that OpCode released its initial version of MAX in the early 90's. Zicarelli's **coll** object resembles the collection in the SmallTalk language. It is an indexed set of rather arbitrary bunches of data. At the time Zicarelli also implemented a message called **refer** that could be sent to the **coll** object. This reference message allowed one to substitute the data in the **coll** with data from another collection. The implementation is very efficient as it involves moving memory pointers to the collection in the reference message.

In the following example, a **coll** containing references is used in a technique for developing elaborate data structures. The basic patch is shown below.



The contents of the **coll eight-programs** is

1, refer program1;
2, refer program2;
3, refer program3;
4, refer program4;
5, refer program5;
6, refer program6;
7, refer program7;
8, refer program8;

The data in the collections **program1 ... program8** contain references to yet other sets of data. In the case of the rhythmic representation as presented here they contain references to the pattern data to be used by a number of different virtual performers.

The mechanism for delivering these patterns to different locations in the program relies on the use of the **route** object. **route** strips off the first element of a list it receives and sends the remaining part of the list to the outlet identified by the first element. So by adding an additional index in front of the collection of references they can be distributed throughout a patch. Example contents of **program5** are shown below.

1, 10 refer pat1;
2, 11 refer pat2;
3, 12 refer pat3;
4, 13 refer pat4;
5, 14 refer pat5;
6, 15 refer pat1;
7, 16 refer pat2;
8, 17 refer pat3;

When dumped into a **route,** the references to patterns are distributed to virtual players 10 thru 17. The patterns **pat1, pat2** contain similarly indexed references to the vectors of the rhythmic representation: the probability vector, the accent vector, the deviation vector, and so on. An example containing the rhythmic pattern vectors is shown below.

```
1,  1  12;
2,  2  127  22  84  127  20  86  127  20  89  127  20  100;
3,  3  2.  1.61   0.87   1.57   0.80  1.47   2.  1.08   1.57   0.98   1.43   0.70;
4,  4  0  5  16  0  5  17  0  5  15  0  7  15;
5,  5  2.  1.19   0.66   1.71   0.91   1.50   0.52   2.  0.66   1.82   0.59   1.19;
6,  6  2.  0.31   0.31   2.  0.35   0.35   2.  0.35   0.35   2.  0.35   0.35;
7,  7  10.  10.  11.  11.  11.  11.  11.  11.  11.  11.  10.  10.;
```

This data structuring technique doesn't turn MAX into LISP but it does go a long way in relieving some of the frustrations I have experienced in manipulating large and varied sets of data.

References relating to this work can be found in the rhythm section of the CNMAT web page - www.cnmat.berkeley.edu.