Macrostructure and Transition in an Algorithmic Composition Environment

Ken Paoli, DMA
College of DuPage
Glen Ellyn, IL USA
paolik@cod.edu

ABSTRACT

Algorithmic composition is applied to a wide variety of composer/machine interactions. The term computer aided algorithmic composition (CAAC) is generally accepted to describe this activity in its various guises. Many software programs and computer procedures that are appealing to a composer yield results that reflect a microstructural composition strategy. Large scale formal events seem to have been a secondary concern for the early contributors to computer music and little has been written concerning macrostructural shaping and transitional linking in algorithmic composition. This paper examines a possible solution for generating larger musical constructs and a method to link them together.

1. INTRODUCTION

More than thirty years after the codification and adoption of the MIDI standard, algorithmic composition has evolved to cover a wide variety of techniques that include different levels of interaction between the composer and the computer. Christopher Ariza offered up a hybrid term to cover the work in this field: computer aided algorithmic composition (CAAC). As this term is generally accepted to cover a multitude of approaches, and it will be used here. Ariza stipulates that “a CAAC system is software that facilitates the generation of new music by means other than the manipulation of a direct music representation” (Ariza, p. 1). Using the criterion of an “equal in, equal out” approach, the mere use of a computer in music is not recognized as computer music. This eliminates notation programs and the use of a hard drive to replace linear (analog or digital) tape in typical recording systems. A CAAC system allows for the manipulation of musical “stuff”, often using numeric representations, logical operations and data manipulation. The MIDI standard with its sometimes frustrating limitations is still rather elegant in its ability to translate musical gestures that the computer can render into sound. While many musicians still reject the use of the computer in any musical endeavor, instead favoring intuitive processes, we should be well past the discussion of whether or not the computer should be used. Like a double-edged sword, the capitalistic producers and performers of commercial music have integrated the computer into most every aspect of production while composers still meet resistance when confessing to use algorithmic procedures. As Phil Winsor stated in *Automated Music Composition,* “The question to ask is not, ‘Is it possible to apply the computer to musical problem solving, but rather, how can it be applied in a meaningful way’…” (Winsor, p. 4).

Copyright: © 2017 Ken Paoli. This is an open-access article distributed under the terms of the Creativ Coons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. COMPOSER/MACHINE INTERACTION

Often, a composer’s initial interaction with the computer is as a substitute for the piano. The computer becomes the composer’s tool bench. This begins a journey of discovery oftentimes altering the composers view of musical style and the use of the musical elements. Achieving the desired musical output requires a detailed problem-solving approach that often challenges the descriptive stylistic “bullet points” that often pass for analysis. The speed of processing available allows the computer to output a vast amount of material for examination and evaluation. The almost instantaneous feedback is invaluable for the evaluation of generated material. This outputting is much faster and more exhaustive than an individual could accomplish *senza macchina*. For some composers, the computer generates the material and the composer uses intuitive means to manipulate the material. For other composers, a higher level of interaction with the machine is desired. The composer seeks additional compositional control through the addition of processing control to the musical parameters.

How are larger compositions created when the computer is generating material? Many computer music authors have advanced the notion of “top-down/bottom-up” aspects of composition. This may be accomplished with or without a computer but for the purposes of this paper computer composition is implied. In his excellent book *Composing Music with Computers*, Eduardo Miranda discusses these approaches. He states that algorithmic software tends to be bottom-up with the computer generating material (via improvisation and/or experimentation) and the composer chooses and stores promising material for future manipulation. This material becomes the basis for creating larger sections. Computer aided composition software often uses a top-down approach: the composer begins with a compositional plan or program with the composer making decisions concerning the criteria for each section (Miranda, p 9). Miranda allows that most composers use both approaches.

Nick Collins cites Miranda stating that the bottom-up approach is driven by the material and that top-down approaches are quite lacking in the history of algorithmic composition. While templates of existing forms may be used (Collins quotes the wonderful Doug Hofstader term “templagiarise”) the interaction of the generated materials and the formal template are “not always so well algorithmized” (Collins, p. 108).

The consensus seems to be that most composers work in both top-down and bottom-up fashions during the course of realizing a composition. However, there is little research available that suggests methods for developing larger musical constructs. Two notable exceptions come from Christopher Dobrian and the doctoral dissertation of Jeremy Leach, whose writings explore methods of larger macrostructural design. Both writers take a hierarchical approach in their designs and of the two, Dobrian uses Schenkerian analysis terms and concepts overtly. A commonality in their thinking is the use of change as a contrast to stable areas, which allows for the construction of larger sections of music.

3. MACRO-STRUCTURAL CONSIDERATIONS

Leach defines a rapid change of state as a “catastrophe” and describes such events as “c-regions”(changing regions). Leach uses and X-Y axis to illustrate these changes of state (Figure 1).

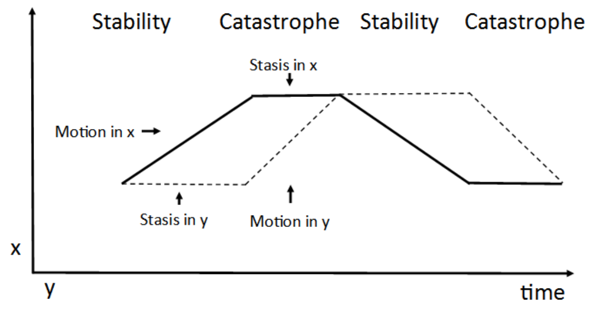


Figure 1. The change behavior for x and y values.

Leach constructs an algorithm utilizing subdivision to expand a “starting unit into a complete composition with structural relationships explicitly defined by a given data set” (Leach, pp. 81-82).

Dobrian uses the terms “stasis and transition” to describe stability and change over time (Figure 2).

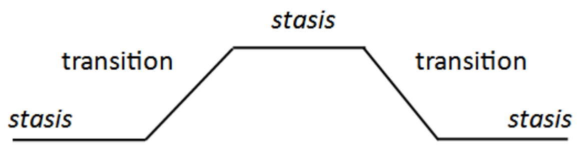


Figure 2. Diagrammatic example of stability and change.

Dobrian states that “a structure is ornamented by departure and return. A static segment is ornamented with a transition to another level and each transition is ornamented with a static segment.” Like Leach, Dobrian also provides an X-Y axis example (Figure 3).

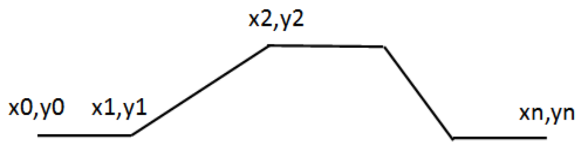


Figure 3. Graphic representation of the Dobrian code.

He further states that experiments in algorithmic composition had focused on “phrase level” events. This is typical of the bottom-up approach which required the composer to use intuitive processes to develop the work and produce a larger composition. This led Dobrian to believe that the composition of larger forms was “fertile terrain for systematic and algorithmic cultivation” (Dobrian, p. 3). He developed an algorithm (in C) to develop contrasting sections:

```
while ((nextx-x)>=minseg{
if (y==next y)
ornstasis ( );
else orntransition ( );
}
```

The code stipulates while next x-x is greater than or equal to the minimum specified segment, if y is equal to next y ornament stasis or else ornament transition. The code is elegant in its simplicity and effective when implemented. Dobrian demonstrates a composition example (in MaxMSP) using this algorithm to control the following musical elements:

Polyphony
Pitch class
Octave registration
Loudness
Orchestration
Tempo
Note Values
Legato

In this example, a probability table for each dimension made stochastic choices for pitch, loudness, duration and timbre. The duration of the composition was determined by the speed of the temporal functions. One could keep the temporal functions and replace the contents of probability table to create divergent compositions.

4. COMPOSITIONAL APPLICATION

Using the code supplied by Dobrian, a real-time patch was derived in MIDIWonk, a graphic programming environment that excels in MIDI data manipulation. Before detailing the implementation of the “Macrostructure” patch a brief explanation of the previously programmed instrument called “OneVoice” is offered. “OneVoice” had yielded good results using probability curves and random processes to generate a monophonic melody. The flow chart for that instrument follows: (Figure 4).

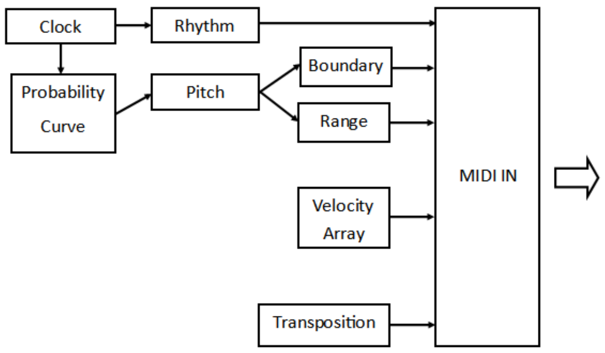


Figure 4. "One Voice" instrument flow chart.

The output of the probability curve module, driven by a clock module, supplies values to the rhythm and pitch module groups. The pitch material is mathematically manipulated (using modulus math) into the (MIDI) range of the desired instrument and the rhythms are quantized into a range from (2-192) representing the range of the MIDI sync clock (based on 24 pulses per quarter note). The rhythms are bounded by user controls to set a range of rhythmic values.

Velocity values are supplied by an array populated with random generated values (in a specified range) to generate variety in the attack velocity of the pitch material. Transposition is bounded plus/minus 6 semitones. The transposition is initiated by a random clock feeding a random integer generator which, in turn supplies the transposition in the aforementioned range.

A legato function is implemented which bypasses the usual MIDI note off event and waits for a MIDI note on message to change the pitch. This eliminates the oftentimes repetitious aspect of random probability distributions.

The velocity and transposition functions were implemented to increase the musicality of the outputted material. Nothing sounds less human or more machinelike than equal velocities for all pitches and the human animal quickly tires of a never-changing, repeated pitch class.

This instrument was converted into a "macro" which allowed for multi-voice polyphonic music to be easily configured and generated. While the results were quite satisfactory, the bottom-up nature of the generated music begged for a method to create larger sections with greater contrast. The "Macrostructure" patch programmed using Dobrian's algorithm utilizes "OneNote" macros as the outputting instruments.

The "Macrostructural" patch utilizes an "Array Slider" module with 10 user programmable faders. A "Step Array" module output moves from one fader value to the next. That movement is dependent upon the result of a "Greater than or Equal to" module which performs the comparison of values from Dobrian's algorithm. A user programmable knob allows for the creation of longer or shorter sections by increasing the value to be compared. The change initiated by the movement through the step array is the tempo of the output. With a system clock controlling the entire patch, it is logical for tempo to be the first agent of parameter change. Since the user can indicate tempi with a wide range of values, a "Slew" module was introduced for a smooth transition between adjacent but disparate tempi (Figure 5).

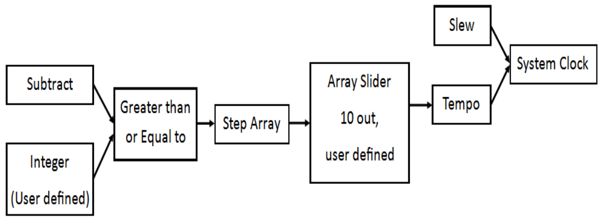


Figure 5. Flow chart for tempo changes to system clock.

Once the change message was sent, the "OneVoice" macros received the velocity, rhythmic value and transposition information. Sending the same information to each macro instrument yielded unsatisfactory results. When the outputting modules were sharing the same velocity and rhythmic information, the result lacked the subtlety and variety associated with human performance. The resulting sameness was obvious and unmusical. A solution that yielded better results was to set up overlapping zones. These zones were much like velocity curves in sampling instruments.

The MIDI velocity range was divided into overlapping zones (in the range from 35-120) and placed in "Step Array" modules. A "Random Integer" module triggered a "Switch" module that sent different zone information to the macro instruments. This yielded a musical result as each instrument in the ensemble could perform in a different velocity range. The same approach was used to send different rhythmic value information to the instrument macros. Overlapping rhythmic zones were stored in an array and randomly selected by a switch module. Pitch material was the same for all "OneVoice" macros but if change was desired the use of existing scales, serial pitch classes or logical sieves could be implemented in the same fashion (Figure 6).

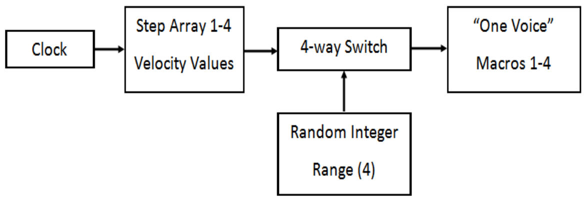


Figure 6. Velocity (and rhythmic) zone diagram.

The Control Panel for four voices (showing Array Sliders, Segment Adjustment, Tempo Count, Tempo Slew, Transposition and Instrument) and the modules for velocity zone processing follow (Figures7 and 8).

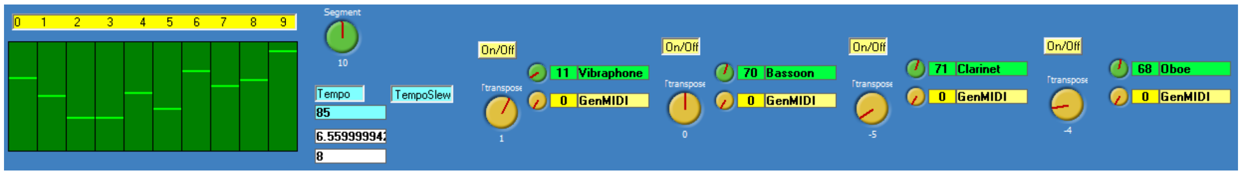
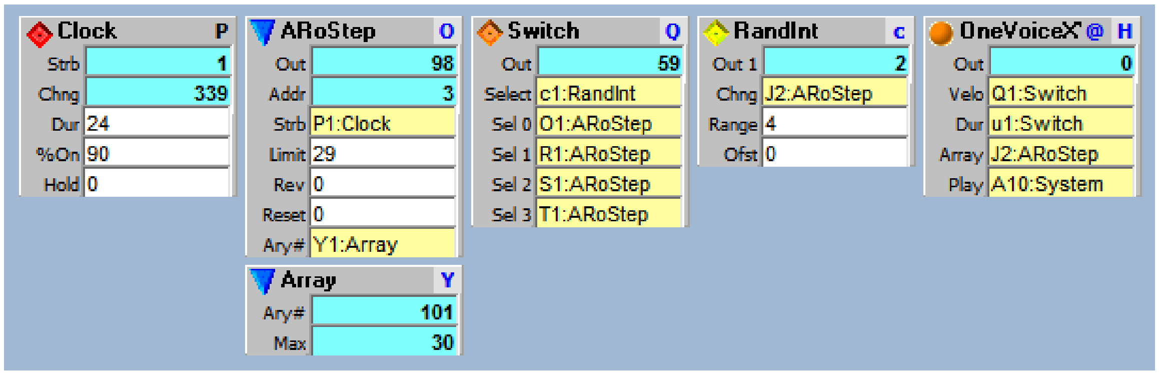


Figure 7. Control Panel for four instrument "voices."



Clock P
Strb 1
Chng 339
Dur 24
%On 90
Hold 0

ARoStep O
Out 98
Addr 3
Strb P1:Clock
Limit 29
Rev 0
Reset 0
Ary# Y1:Array

Array Y
Ary# 101
Max 30

Switch Q
Out 59
Select c1:RandInt
Sel 0 O1:ARoStep
Sel 1 R1:ARoStep
Sel 2 S1:ARoStep
Sel 3 T1:ARoStep

RandInt c
Out 1 2
Chng J2:ARoStep
Range 4
Ofst 0

OneVoiceX @ H
Out 0
Velo Q1:Switch
Dur u1:Switch
Array J2:ARoStep
Play A10:System

Figure 8. Velocity (and rhythmic) processing modules.

The output of the "Macrostructure" patch yielded excellent results and allowed the generation of sections of varying size. By using the same tempo shape with the array sliders and changing the segment size, the *Ursatz* of larger sections could be used to define the formal shape of smaller sections.

5. TRANSITIONS

As often happens, a successful implementation leads to new expressive demands requiring new programming solutions and so it was with the "Macrostructure" patch: a new concern arose. While the sections were longer and the contrasts provided episodic formal division, the changes were often abrupt and lacked thematic connection. This problem is also typical of music generated in a bottom-up fashion. Interesting material is followed by (hopefully) more interesting material, but little or no connection exists between them. This led to the examination of possibly deriving transition sections to link the larger constructs. One would have to forego the real-time output of the patch since a transition needed both the material just heard and the material to be heard. This, like non-computer composition, would need to be accomplished "out-of-time." The sectional material would be recorded as would the transition. The transitions would then be inserted into the recorded output of the patch. Music Wonk provides real-time processing of the material and allows for MIDI output to be routed to and recorded in any standard DAW. This is accomplished via a software MIDI interface like loopMIDI.

As is the case with macrostructures in CAAC, there is little written concerning transitions. Jeremy Leach states that notes nearer to a transition have greater significance for the listener (Leach, p. 99). He based this on the perceptual work and experiments of Diana Deutsch. The notion is that listeners remember information from the beginning of a transition but middle events are not as prominent to the listener. Leach presents additional arguments: transitions tend to be shorter than the surrounding stable musical material and transitions can also model hierarchical structuring. Using examples from visual art and natural phenomena in biology, Leach develops a model using a "hierarchical backbone upon which to structure resulting compositions" (Leach p.102). The size of the element at a specific level will determine the number of "smaller c-regions" (changing regions)" into which a c-region should be divided. This approach to structure again invokes the theoretical concepts of Schenker. While appreciating the conceptual notions of Leach, his model is part of a larger programming construct and implementation to the existing macrostructural patch already in use was deemed impractical.

There is no lack of examples of transitions in music literature. The bridge theme that connects two thematic areas with different tonal centers in a sonata might be the most familiar. The connecting area is an example of harmonic instability often accompanied by irregular and asymmetrical constructs in melody, rhythm and phrase. A transition can be as simple as a prolonged element that is redefined in the context of a following section (a common tone), a transition with completely new material or a transition that, as Shoenberg stated, contains the "liquidation of motival characteristics" (Schoenberg, p.179). The notion of motivic liquidation infers the fragmentation and lessening of motivic elements across a time span. It would be followed by the new section. In the context of the macrostructural patch it was thought that attempting to overlap material from two adjacent sections would be a type of "liquidation" of material. At the same time, it would preserve the stochastic aspects of the program.

To compare the materials from two varying sections and create a transition section, "Read/Write Array" modules are used to accept mirrored output data from the "OneVoice" modules. This contains the material for any two sections that could be linked by a transition. Three types of data are recorded: note data, velocity, and rhythmic values. The contents of the "mirror" array is divided into two groups of three data files. One group contains the pitch, rhythm and velocity of Section 1 and the second group contain Section 2 material. These files are loaded into the array group for processing and recording (Figure. 9).

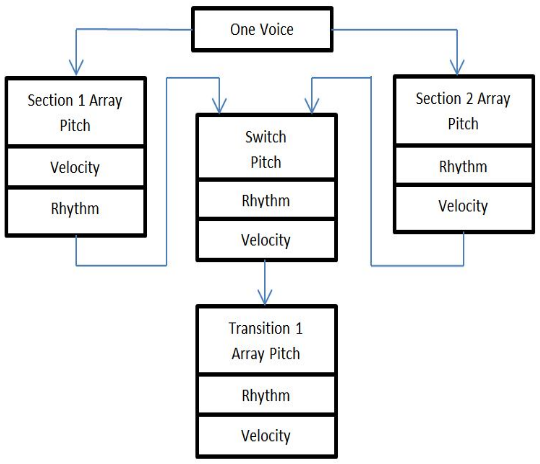


Figure 9. Transition flow chart using material from two sections.

It was conjectured that more velocity choices would add to the variety of the transitional section. To this end, arrays that provide values for crescendos, decrescendos, and a combination of the two were added. Each array was input to a "Switch" module which is toggled by a random integer. The number of switch choices is parameter- dependent. The pitch and rhythm switches have two choices (one for each array); the velocity switches have four choices (velocity values of the original velocity material and three additional velocity curves). Upon receiving a clock message at the strobe input, the switch will activate and start the transition with material indicated at the switch output. This portion of the patch is displayed in Figure 10.

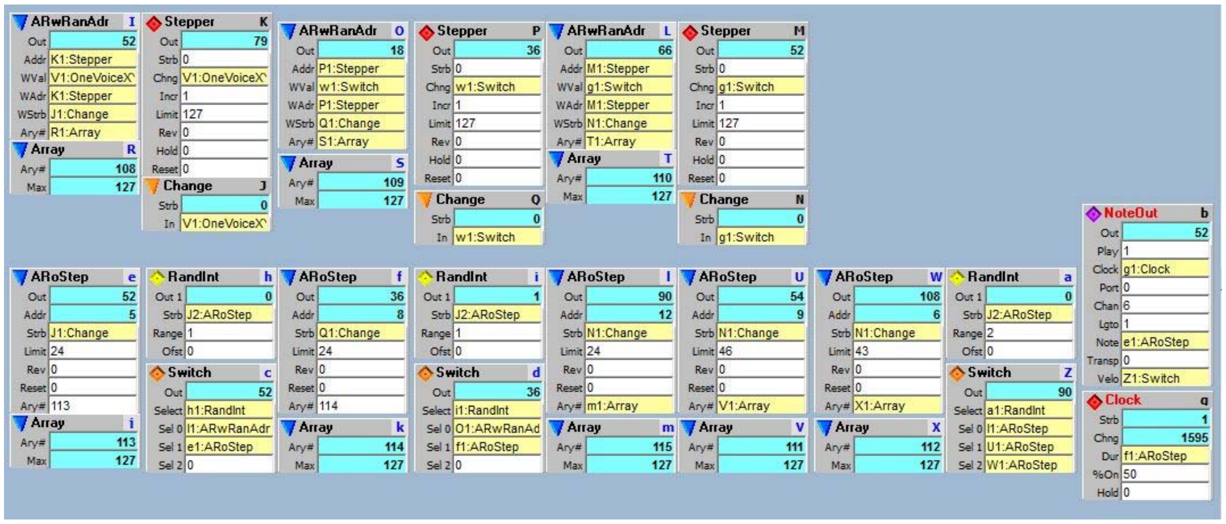
For the initial test, the outputted array material from one instrument was recorded as MIDI data in a DAW program. The track was inserted into a previously recorded session of the source sections. The transition material was compared to the sectional material that was meant to precede and follow the transition. Besides the array manipulation, further "out of time" work involved making space in the original track for the transition material. The additional velocity curves helped produce a satisfying dynamic shape to the transition and, while the material from the source arrays were not comingled, the result of the combination of the source material worked quite well.

Subsequent test runs of the same material yielded different switch choices in keeping with the stochastic nature of the original patch. Some runs placed the second section material before the first which created a "foreshadowing" of the larger section to come with a sonic "reminder" of the section just heard. This was deemed the most effective and musically satisfying test outcome. Using a random output from the Transition array was most unsatisfactory; the connection to the source material became unrecognizable. While the material was comingled the effect of this yielded no sense of transition at all. Adding the additional "OneVoice" instruments greatly increased the number of arrays necessary to accommodate the musical elements. It was decided to place the Section and Transition arrays in macros to declutter the work space and streamline the patch. Macros for the velocity zone arrays and the rhythmic value arrays were also implemented. The process of placing multiple tracks of transition material was time-consuming and the best results were obtained after compositional manipulation of the generated data. The overall complexity of the "Macrostructure" patch was increased but the ability to add transitions, although "out-of-time", was welcomed. Recognizing that this is a rather singular approach to a generative transition, it still provides a method to access the processing power of the computer to provide material for the composers' examination and manipulation.

6. CONCLUSIONS

Historically, formal considerations have not been at the forefront of computer music concerns. For the pioneers in both electronic and computer music, perhaps the burdensome process of generating sound and processing data precluded concerns with larger formal considerations. With the increase in the speed of processing and the pervasive nature of computer accessibility, computer musicians are able to embrace more than the bottom-up approach to algorithmic composition. Creating larger algorithmically generated works with macrostructural control is obtainable when a "top-down" approach uses one element to trigger changes in multiple elements. Tempo is a logical candidate to initiate the change since generally one system clock controls most programs. The resultant sectional contrast fits the notion of stability followed by transition (or catastrophe) as found in the writings of Dobrian and Leach. Schenkerian concepts are espoused by both authors indicating the desire to use hierarchical, organizational concepts in the design of their algorithms. Hierarchical structures are relevant even with stochastic organization of the musical elements. The Dobrian algorithm yields excellent results and by changing the size of sections, a single formal design can be used to generate both background and foreground events. Leach uses a hierarchical backbone that links background and foreground levels and is the basis for the compositional organization in his program. Although several years have passed since the writings of Dobrian and Leach were published, "bottom-up" programs remain an excellent entry point for many composers interested in algorithmic composition.

The exploration of macrostructure and transition discussed here were driven by the authors' desire to expand



```
ARwRanAdr I        Stepper K              ARwRanAdr O        Stepper P          ARwRanAdr L        Stepper M
 Out   52           Out    79              Out   18           Out    36          Out   66           Out    52
 Addr  K1:Stepper   Strb   0               Addr  P1:Stepper   Strb   0           Addr  M1:Stepper   Strb   0
 WVal  V1:OneVoiceX Chng   V1:OneVoiceX    WVal  w1:Switch    Chng   w1:Switch   WVal  g1:Switch    Chng   g1:Switch
 WAdr  K1:Stepper   Incr   1               WAdr  P1:Stepper   Incr   1           WAdr  M1:Stepper   Incr   1
 WStrb J1:Change    Limit  127             WStrb Q1:Change    Limit  127         WStrb N1:Change    Limit  127
 Ary#  R1:Array     Rev    0               Ary#  S1:Array     Rev    0           Ary#  T1:Array     Rev    0
Array R             Hold   0              Array S             Hold   0          Array T             Hold   0
 Ary#  108          Reset  0               Ary#  109          Reset  0           Ary#  110          Reset  0
 Max   127         Change J                Max   127         Change Q            Max   127         Change N
                    Strb   0                                  Strb   0                              Strb   0
                    In     V1:OneVoiceX                       In     w1:Switch                      In     g1:Switch

AroStep e          RandInt h              ARoStep f          RandInt i          ARoStep l          ARoStep U          ARoStep W          RandInt a           NoteOut b
 Out   52           Out 1  0               Out   36           Out 1  1           Out   90           Out   54           Out   108          Out 1  0            Out    52
 Addr  5            Strb   J2:ARoStep      Addr  8            Strb   J2:ARoStep  Addr  12           Addr  9            Addr  6            Strb   J2:ARoStep   Play   1
 Strb  J1:Change    Range  1               Strb  Q1:Change    Range  1           Strb  N1:Change    Strb  N1:Change    Strb  N1:Change    Range  2            Clock  g1:Clock
 Limit 24           Ofst   0               Limit 24           Ofst   0           Limit 24           Limit 46           Limit 43           Ofst   0            Port   0
 Rev   0           Switch c                Rev   0           Switch d            Rev   0            Rev   0            Rev   0           Switch Z             Chan   6
 Reset 0            Out    52              Reset 0            Out    36          Reset 0            Reset 0            Reset 0            Out    90           Lgto   1
 Ary#  113          Select h1:RandInt      Ary#  114          Select i1:RandInt  Ary#  m1:Array     Ary#  V1:Array     Ary#  X1:Array     Select a1:RandInt    Note   e1:ARoStep
Array i             Sel 0  I1:ARwRanAdr   Array k             Sel 0  O1:ARwRanAd Array m           Array V            Array X             Sel 0  l1:ARoStep   Transp 0
 Ary#  113          Sel 1  e1:ARoStep      Ary#  114          Sel 1  f1:ARoStep  Ary#  115          Ary#  111          Ary#  112          Sel 1  U1:ARoStep   Velo   Z1:Switch
 Max   127          Sel 2  0               Max   127          Sel 2  0           Max   127          Max   127          Max   127          Sel 2  W1:ARoStep  Clock q
                                                                                                                                                              Strb   1
                                                                                                                                                              Chng   1595
                                                                                                                                                              Dur    f1:ARoStep
                                                                                                                                                              %On    50
                                                                                                                                                              Hold   0
```

Figure 10. Section 1 to Transition Array module.

beyond “bottom-up” approaches and find satisfactory methods to approach larger sections or entire pieces. When using generative programs, logical aural connections between disparate (or similar) sections is generally absent. Designing a control process to yield a method of transitioning was a logical “next step.”

It should be mentioned that today much emphasis is placed on sample manipulation and digital processing as a working method to generate materials and whole compositions. This seems to be another “bottom-up” approach but it does resemble the crystallization analogy employed by Varese when discussing form. Future sound events become a resultant of current event manipulation. Form is a resultant not a “primary attribute” (Varese, p. 16). Regardless of the method employed to generate material, the shaping of sound across time is a concern for most composers and macrostructural considerations still remain an area for exploration and compositional research.

Transitions seem to be mentioned less than macrostructure in computer music writings. It is possible to generate formal sections that abut each other; however, it sometimes creates too abrupt a change. A transition can make changes between sections smoother and more musically pleasing. Since transitions involve a connection of music already heard to music yet to be heard, a composer must sacrifice the “in time” generation of a real-time program and return to an “out of time” compositional solution. This involves multiple arrays controlling multiple musical parameters as well as recording and manipulation of material. There is a fair amount of manipulation to generate the material and place it in the composition. The results obtained by this method would seem to make that effort worthwhile.

7. REFERENCES

[1] Ariza, Christopher. "Navigating the Landscape of Computer Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." Cambridge: MIT OpenCourseWare, 2010. 1-9.

[2] Collins, Nick. "Musical Form and Algorithmic Composition." *Contemporary Music Review* February 2009: 103-114.

[3] Dobrian, Christopher. "Algorithmic Generation of Temporal Forms:Hierarchical Organization of Stasis and Transition." *International Computer Music Conference.* 1995.

[4] Dunn, John. MusicWonk. Algorithmic Arts.2003.

[5] Leach, Jeremy L.. *Algorithmic Composition and Musical Form (*Doctoral Dissertation*).* University of Bath, 1999.

[6] Miranda, Eduardo Reck. *Composing Music with Computers*. Ed. Francis Rumsey. Burlington: Focal Press, 2002.

[7] Schoenberg, Arnold. *Fundamentals of Musical Composition*. Ed. Gerald Strang. London: Faber and Faber, Ltd., 1967.

[8] Varese, Edgard and Chou Wen-Chung. "The Liberation of Sound." *Perspectives of New Music* Vol. 5.No. 1 (1966): pp.11-19.

[9] Winsor, Phil. *Automated Music Composition*. Denton: University of North Texas Press, 1992.