Composition Design System: A Functional Approach to Composition

Keith Hamel  Bruce Pennycook  Bill Ripley  Eli Blevis

Computer Music Facility
Department of Music, Queen's University
Kingston, Ontario
CANADA, K7L 3N6

Abstract
This paper presents a discussion of the design and implementation plans for the Composition Design System (C.D.S.) that is currently in development at Queen's University. Initially, we discuss the view of the compositional process that underlies C.D.S. and the basic design criteria of the system. Next, we present a strategy for representing our notion of composition and the compositional model that defines it. Finally, implementation details and possible extensions to the system are discussed.

Objectives

Background
Computer languages for the description of musical processes continue to pose one of the most intriguing challenges to composers and software designers alike. The difficulties stem from the complexity of music itself and from a general conflict in programming language design - procedural versus functional. The literature abounds with attempts to reconcile specificity - explicit lists of compositional or synthesis data, with generality - formal descriptions of musical processes. Both approaches are further confounded by the essential ingredient of time which few programming environments support.

Many composition systems tend toward specialized tools for the production of specific musical styles or genres, for example Trux's POD system (Trux, 1985). Another approach has emerged from research in formal grammars as applied to musical discourse (Roads, 1979). (Holtzman, 1981) although no complete implementation of a grammar-based system is known to us.

Several strategies based on functional programming techniques have served as valuable models for C.D.S. The relative advantages of LISP were asserted in "Machine Tongues VII" (Kornfield, 1980). At the same time, an intriguing approach to jazz improvisation was described by C. Fry (Fry, 1980) using routines developed on a Symbolics LISP Machine. The most comprehensive LISP system is Formes (Roder and Coti, 1984), which integrates control of a variety of synthesis mechanisms within an object-oriented programming environment.

C.D.S. resides in the more modest environment of the Macintosh and MIDI and at this time does not include synthesis specification beyond MIDI accessible descriptions. However, like Formes, much attention has been given to the notion that composers should be able to construct work-spaces which best suit their aesthetic designs and, according to Rodet and Coti, "... a model should not be a portrait of a particular sound or note but should be a general representation of a process as possible..." (ibid. 32-33). In our case, we have modified this notion somewhat such that the representation of the musical structure can be viewed as distinct from events, notes, phrases and even the entire composition.

CDS Premise
The premise that underlies Composition Design System (C.D.S.) is that music composition can be expressed as a functional representation of musical devices or transformations. A music composition can be described in terms of music gestures\(^1\), transformations of those gestures and the relations that exist among those transformations. This premise suggests that the creation of a composition involves encoding music relations. To a large extent, it is the perception (either consciously or intuitively) of these relations along with the comprehension of the strength of compatibility\(^2\) amongst gestures that constitutes our understanding of a musical work. From this perspective, the surface level of a composition is less significant than the structure of transformations or functions that act upon the music gestures. C.D.S. provides an environment for the creation of compositions from a description of the compositional design.

The following example illustrates our view of the compositional process. Although the two music fragments below are very different, they can be viewed as two instances of the same compositional design (called an abstract composition) in the relation that exists between motifs, in each case, is identical.

1Music gestures refer to any collection of music materials: notes, chords, pitch sets, timbres, rhythms etc.
2The term "compatibility" (which is equivalent to "conformant relationship") is borrowed from Leonard Meyer (Meyer, 71).

1987 ICMI Proceedings 33
2. Flexibility
The purpose of C.D.S is to generate musical examples derived from the musical relationships specified by the composer. To facilitate the formulation of these relations, functions are designed so that they can be combined with other functions and can be applied to any level of the musical hierarchy. This results in a flexible system, one which does not restrict the composer to any given musical style or to any one computational process. This flexibility, however, requires that the user has an explicit understanding of what relations will exist in the composition. If these relations can be clearly expressed, then the system will generate the musical events represented by these relations. This means that C.D.S allows the user to abstractly describe artistic or music ideas which can subsequently be generated as a musical representation.

The system can be used in a wide variety of ways: a composer can specify the relations to the point of controlling virtually every note, rhythm and harmony, or specify general controls and utilize functions which generate materials directly or derive new gestures from previously stored material.

3. Extensibility
The system is designed so that it is infinitely extensible: an individual composer can build functions from a library of previously-defined functions and add these new definitions to the library. Over a period of time, increasingly higher level functions can be described and the operation of the system can become highly specialized.

Strategy
The strategy for carrying out the system's design based on the objectives described above was to focus on the following three areas: the method by which the composition would be represented, the computational model underlying that representation, and the graphics interface to this representation.

Composition Representation
It appeared that the most effective method of representing our conception of music composition was to adopt a functional language model. Such an approach allows the abstract composition to be completely specified without reference to specific musical materials (called primary gestures). Essentially, an abstract composition is a functional representation of the processes or musical devices which operate upon these gestures. An abstract

By function we mean a musical device or transformation encoded as a LISP program segment.

Design Criteria
In order that C.D.S be useful for a wide variety of compositional activities, the following criteria were deemed essential:
1. that a hierarchical representation of music structures be available,
2. that the system have the flexibility to support a wide variety of compositional styles,
3. that a mechanism by which the system could be extended and modified to suit individual composers be included,
4. that the computational representation satisfy certain criteria of a well-formed program.

1. Hierarchical Representation
Musical constructs are normally considered to form hierarchical structures: a phrase can be thought of as a collection of motifs, a section a collection of phrases and so on. Therefore, C.D.S. allows the composer to specify his own musical hierarchy during the composition process. The levels of the hierarchy may be described with pre-defined labels (i.e. event, motif, phrase, section), or with user-defined names. A hierarchical representation is crucial because it facilitates the transformation of music gestures at various levels in the composition.
composition may consist of varying densities of functional layers (called compositional streams) which are associated to one another in relative time.

In order that a functional model of composition be sustained, functions are designed so that they are able to operate upon music gestures at any level in a composition hierarchy. For example, the function transpose can be applied to events, motifs, sections, or any other hierarchical construction. In our functional model of composition, it is essential that all functions have the capability of being freely chained to other functions. This is an important component of the design not only because it allows the expression of complex music transformations, but also because it facilitates the construction of extensions to the function library.

When a user is designing a composition (i.e. defining an abstract composition), functions can be grouped to embrace increasingly larger descriptions of the piece. This facility allows the newly designated group to be passed to other functions. Since each level in the hierarchy can have different properties associated with it, it is possible to perform operations on the higher level representation that were previously undefined. For example, if a number of functions at the motif level are grouped as a phrase, information about harmonic content, pitch range etc. could be included as properties of the phrase and accessed by functions operating on this phrase.

Computational Model

General Description of the Model

The notions of gestures and devices (i.e. musical processes or transformations) are represented in our computational model of C.D.S. as data-objects and functions respectively. However, the representation of time (horizontal relations) and density (vertical relations) require special treatments.

Our purpose in this model is to separate what is concrete about a musical composition (particular gestures) from what is abstract (form). This distinction allows the abstraction to be applied to a variety of gestures yielding a collection of compositions all contrasting to the same abstraction. By analogy, we wish to support a view of composition which separates the "nouns of music" from the "verbs".

Primary Gesture

Primary gestures are the basic thematic material of a musical composition and are represented as a hierarchical structure.

Definition 1: primary gesture

We represent primary gesture as a rooted tree with all nodes labelled uniquely and with arcs ordered at each node. A group G is a non-terminal node of P (a node of outdegree n, n>0). An event E is a terminal node of P (a node of outdegree 0). (Refer to Figure 3.)

Group and Properties

A primary gesture may be organized into groups of associated sonic events such as motifs, phrases or sections.

Definition 2: group properties

Associated with each group G of P is a set of properties G* containing elements that are ordered pairs of the form <Phrase,Prop> where Phrase is an arbitrary string and where Prop is an arbitrary value. G* always contains at least one element, the pair <group-label>,s where s is an arbitrary string such as 'motif', 'phrase' or 'section'. In any given application of this model the element <group-label>,s may be used to infer which other elements occur in G*.

(Refer to Figure 3.)

Events and Properties

At the bottom of the hierarchy of any particular primary gesture are sonic events such as chords, notes, rests, or timbral specifications. What is central to the model is that each event is associated with a duration.

Definition 3: event properties

Associated with each event E of P is a set of properties E* containing elements that are ordered pairs of the form <Phrase,Prop> as stated in definition 1. E* always contains at least three elements:
1. <event-label>,s, where s is an arbitrary string such as 'note', 'chord' or 'timbre'.
2. <duration>,d, where d is a positive integer.
3. <start-time>,t, where t is initially the integer 0.

In any given application of this model the element <event-label>,s may be used to infer which other elements occur in E*.

\[
\begin{align*}
G_1 & \xrightarrow{<\text{group-label},\text{phrase}>,...} \\
G_2 & \\
G_3 & \xrightarrow{<\text{group-label},\text{motif}>,...} \\
S_1 & \\
S_2 & \\
S_3 & \\
S_4 & \\
S_5 & \\
E_1 & \xrightarrow{<\text{event-label},\text{note}>,..., <\text{duration},>,..., <\text{start-time},t>,...} \\
\end{align*}
\]

Figure 3: Primary Gesture.
Devices
The devices (or transformations) as they are understood in music are represented as functions which derive new gestures from initial primary gestures.

Definition 4: Duration function
The duration function d of a primary gesture G is the integer sum of the duration fields of all of the event (i.e., terminal) nodes in G.

Definition 5: Device properties
Let P* be the domain of primary gestures. We represent a device as a function  from P* to P* and let F* be the domain of all such functions f.

Compositional Streams
Devices are organized into compositional streams which group derived primary gestures into a sequence to be performed in time. A compositional stream is in fact a function formed from devices and applied to a single primary gesture.

Definition 6: Compositional stream
A compositional stream is an ordered set S of devices  f_1,...,f_n with n > 0 notated [f_1,...,f_n]. An interpretation S of S with respect to a primary gesture P, denoted S(P), is an ordered set of elements of [f_1,...,f_n] given by the equation  f_1(P),...,f_n(P).

Abstract Composition
An abstract composition is a network of compositional streams ordered according to time and density (i.e., compositional streams per unit time). More precisely, it is a function from primary gestures to derived primary gestures organized in time.

Definition 7: Abstract composition
We represent an abstract composition as a (weakly connected) directed graph F. Each node D (called a density) in F has associated with it a compositional stream S, and an integer t, denoting a start time. Furthermore, each D in F has outdegree 0 or 1. If the outdegree of any D is 0 then its start time t is 0 and it is called a left-most compositional stream. If the outdegree of any D is 1 then its outgoing arc is associated with some f_1, 0 < k < n, of the compositional stream S.
We note a concrete composition as a table:

<table>
<thead>
<tr>
<th>Primary Gesture</th>
<th>Abstract Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>[f₁, ..., fₖ, ..., fₖ]</td>
</tr>
<tr>
<td>Density</td>
<td>[F₁, ..., Fₖ]</td>
</tr>
<tr>
<td>Time</td>
<td>[F₁, ..., Fₖ]</td>
</tr>
</tbody>
</table>

Figure 6 - Graphic Representation of an Abstract Composition

The following example maps a music composition to a concrete composition:

<table>
<thead>
<tr>
<th>Primary Gesture</th>
<th>Abstract Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>P = (g₁ g₂)</td>
<td>rep seq [seq trseq]</td>
</tr>
<tr>
<td>Density</td>
<td>limit [seq] (Compositional Streams)</td>
</tr>
<tr>
<td>Time</td>
<td></td>
</tr>
</tbody>
</table>

LISP Definitions:
- rep = λ. P (P)
- seq = λ. P (sequence up1 P)
- trseq = λ. P (retrograde(sequence down1 P))
- limit = λ. P (sequence down15 P)
- trseq = λ. P (sequence down12 (fragment 2 tr1))

Graphics Interface
In order to provide a more efficient editing environment, a graphic representation of the abstract composition forms the user interface of C.D.S. Functions can initially be defined with text, then converted to a icon which can be placed on a density/time graph. These density/time graphs constitute a visual description of an abstract composition. Time coordination between compositional streams is also represented on this graph. Selecting a symbol causes its composite functions to be displayed, thus making the various levels of the compositional hierarchy accessible. An interactive mechanism by which functions are selected and grouped to form new hierarchical levels in the abstract composition is included in the interface design. The primary gesture (which may include several distinct music fragments) is represented symbolically to the left of the graph.

It is important to state that the graphic display is not a representation of music materials, but rather a representation of the formal design of the composition. Given that time is relative to other events, and given that the primary gesture (which may vary in size and complexity) is distinct from the abstract composition, the graphic representation can display only relative timing information. This limitation is analogous to a description of a compositional form which is independent of the total duration of composition.

Implementation

Hardware/Software
We decided to implement C.D.S. on the Apple® Macintosh™ personal computer for the following reasons:
1. We were interested in designing the system for a personal computer so that it would be accessible to many users.
2. We wanted to take advantage of the Macintosh's graphics capabilities.
3. The proliferation of music software (i.e. sequencers, notation software etc.) for the Macintosh provides many options for extensions to the C.D.S. user's working environment.

Le LISP® was chosen as the development software for C.D.S. because the MIDI extensions (MIDI LISP) simplify the conversion to and from sound.

C.D.S. Components

Data Library
The Data Library contains many of the musical constructs (notes, intervals, chords, scales, rhythmic groupings etc.) that might be desired by a user. All pitches and intervals are encoded in their correct
enharmonic form using Clements’ numbers.\footnote{This is our term to denote the encoded pitch according to the system developed by Peter Clements.} (Clements, 1986). In addition to the pre-defined structures, new definitions may be added by combining existing structures.

**Function Library**

Conceptually, there are three types of functions available in C.D.S.:

1. functions that generate new music gestures directly,
2. functions that generate new music gestures from gestures passed to them,
3. functions that generate new music gestures through analysis or extraction from previous gestures.

The first category consists largely of mathematical functions such as Markov chains or other random procedures.

The second category contains the majority of C.D.S. functions and can be sub-divided into three classes: melodic, harmonic, and rhythmic. These functions generally search for specific properties in a gesture (i.e. a melodic function will normally look for the pitch property, i.e. Clements’ number of a note).

The third category includes two types of extraction functions. The first of these, get-material, allows the composer to use gestures from any previous section. The position of the gesture must be specified (e.g. the second motive in the third phrase). The second function, extract-material, is much more powerful. The user can request that any material with certain characteristics be extracted. For example, one could ask for "any five note fragment that contains a G as well as an inversion of Motive A."

The function finds all occurrences of this material and returns a list of them.

As mentioned earlier, all functions have the ability to be combined with other functions in any fashion, e.g. harmonize (inverse) (augmentation(A)) where A is a primary gesture. If a function is unable to operate upon the gesture (perhaps because the required property is not present) it passes the gesture unaltered and the chain continues unbroken.

**Data Structure**

Each hierarchical level is a music gesture may have unique properties. Properties are easily represented in LISP as property lists. When functions are applied to a high (i.e. non-terminal) hierarchical level, C.D.A. employs data structure functions to search the hierarchy until the correct properties are found. For example, applying the function invert to a phrase causes the phrase hierarchy to be searched and all pitches at the event level to be inverted.

**Using the System**

To use C.D.S., the composer can begin by either defining the abstract composition or by specifying the primary gesture to be used. The primary gesture can be supplied either by indicating a hierarchy of events and groups, or by providing MIDI input. Any number of properties can be associated with the hierarchical levels of the primary gesture.

The abstract composition is created by applying functions (i.e. musical transformations) either to the primary or derived gestures. These functions are identified using words or graphic symbols and placed on the structure chart to form a compositional stream. There is a mechanism by which compositional streams can be associated in time with the completion of a function within another compositional stream. Portions of the abstract composition (i.e. the series of functions) can be collected to form a group (either pre-defined, such as phrase, or user-defined) and properties can be associated with this group. The newly designated group can be used in the definition of other functions.

When the abstract composition has been completed, a concrete composition is generated by evaluating the abstract composition and the primary gesture. The composition in played via MIDI. After audition, modifications can be made to both the abstract composition and the primary gesture.

**Discussion**

**Present State of the Project**

At present, the system design, data structure and computational model have been completed, and many of the functions have been implemented. The user interface and control mechanisms (mainly between the abstract composition and the creation of concrete compositions) are still in development.

**Future Extensions**

We would like to provide a mechanism for building high-level functions from a collection of function fragments. This feature would entail the definition of new functions using a combination of conditional statements and would become a more powerful method for extending the function library.

With respect to abstract compositions we want to consider higher-order functions from compositional streams to compositional streams. We expect these will allow us to express such things as constraints between various functions in a particular compositional stream.
Another extension will be the capability to pass messages in the form of functions among compositional streams.

Conclusion

C.D.S. is an attempt to build a composition system by describing a functional view of the compositional process, devising a compositional model from this description and implementing this model in a graphics environment. The flexibility and extensibility of the C.D.S. design should permit a wide variety of compositional uses.

Acknowledgements

This work has been supported by a Canada Council Integrated Media Grant. This support is gratefully acknowledged.

Special thanks to Troy Spez for his assistance in the design and implementation of this system.

Le „Lisp is a register trademark of AC“: Informatique.

Apple is a registered trademark, and Macintosh is a trademark of Apple Computer, Inc.

References


