The Development of Context Sensitivity in the Midiforth Computer Music System

Bruno Degazio, M.Mus.
Department of Phono/Electric Arts
Ontario College of Art
Toronto, Canada

ABSTRACT

This paper reports on the development in the MIDIFORTH computer music system of context-sensitive editing features, such as the ability to highlight MIDI events based on their position within a melodic or rhythmic pattern, and on pitch or other relationships to surrounding events. The software mechanism that implements this is discussed, and some examples of the musical desirability of such features are presented.

BACKGROUND

Many microcomputer-based MIDI-oriented computer music systems now exist and are readily available. Almost all of these allow the processing of MIDI event data in a fashion analogous to that found in simple word processing software. The ability to insert, delete and modify individual characters (i.e. MIDI events) is always present, while more sophisticated software will allow editing based on the principle of contiguity - the ability to group characters or MIDI events based on their proximity to one another. The next level of complexity consists of the ability to search for and modify events based on some intrinsic or implied characteristic - (Buxton et al., 1981) calls this grouping by local attributes. An example is the ability to search for and modify all events consisting of a particular MIDI pitch or velocity (intrinsic characteristic) or of a certain duration (implied characteristic). In the language sphere, this corresponds to the ability to search for and modify individual words, such as the "search-and-replace" feature found in most word processors. The most common extension to this sort of editing involves the implementation of relational schemes; for example, the ability to edit all notes 'greater than' (i.e. of a higher pitch) some particular MIDI pitch value, or to restrict editing to events falling only within a certain range (e.g. between middle C and F#).

403
However, few computer music systems - and so microcomputer-based systems except for MIDIFORTh - have investigated the development of editing based on what have been called contextual attributes (Buxton et al., 1981b). Earlier efforts, such as the editing features of the Structured Sound Synthesizer Project (from which some of the terminology used in this article derives), were sketchy and exploratory. The development of such features in the MIDIFORTh computer music system has experienced several iterations, made possible by the extensible nature of the underlying language and operating system, Forth.

THE MIDIFORTh FILTER

The first step in the development of context sensitive score editing in MIDIFORTh was the implementation of a flexible system to allow editing based on a combination of local characteristics. The MIDIFORTh filter originated in the SSSP concept of scope, "a subset of the musical events of a score, selected according to some criteria" (Buxton 1979); that is, the restriction of some useful operation to a musically distinct set of data. In addition to the common restriction of specifying an operation within a temporal boundary (i.e. based on the principle of contiguity), MIDIFORTh allows the following constraints:

<table>
<thead>
<tr>
<th>Pitch:</th>
<th>Equal to:</th>
<th>A fixed value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume:</td>
<td>Not equal to:</td>
<td>The next note.</td>
</tr>
<tr>
<td>Duration:</td>
<td>Greater than:</td>
<td>The previous note.</td>
</tr>
<tr>
<td>Articulation:</td>
<td>Less than:</td>
<td>The next tagged note.</td>
</tr>
<tr>
<td>Pitch Bender:</td>
<td>Modulo:</td>
<td>The previous tagged note.</td>
</tr>
<tr>
<td>After Touch:</td>
<td>Not modulo:</td>
<td>User defined format.</td>
</tr>
<tr>
<td>Continuous Controller:</td>
<td>On a specified beat.</td>
<td></td>
</tr>
<tr>
<td>Switch Controller:</td>
<td>Not on a specified beat.</td>
<td></td>
</tr>
<tr>
<td>Pitch Change:</td>
<td>Member of a defined mode.</td>
<td></td>
</tr>
<tr>
<td>Note Number:</td>
<td>Not a member of a defined mode.</td>
<td></td>
</tr>
<tr>
<td>Position In Bar:</td>
<td>In the search pattern.</td>
<td></td>
</tr>
<tr>
<td>Tag:</td>
<td>Not in the search pattern.</td>
<td>Set.</td>
</tr>
<tr>
<td>Pitch Interval:</td>
<td>Reset.</td>
<td></td>
</tr>
</tbody>
</table>

To following note:

| Volume Interval (to following note): | |
| Duration Difference (from following note): | |

**Table 1**

ICMC Proceedings, 1988 404
In addition, multiple selection criteria (up to three) may be employed by combining with the standard Boolean AND, OR and XOR operators; thus, for example, a transposition operation may be restricted to all notes such that:

IF volume is greater than 72 AND IF duration is less than 48 (i.e., a quarter note) OR IF position is on a specified beat THEN transpose up an octave.

A simple graphic editor is provided for the purposes of altering filter parameters (figure 1).

```
SET TELTERING
IF pitch is in the search string.
IF volume is greater than or equal to the previous note.
IF pitch interval is equal to a constant (or)
filtering is on.
"EXIT" to exit.
```

(Figure 1)

Almost all operations within MIDIFORTH are filterable; the few excluded are those that could conceivably have no use, e.g., the DUR? operator, which returns the total duration of a portion of the phrase. Filterable operations include:

| TRANSPOSE | INVERT |
| MODULATE | FAMP |
| CHAB | RANDOMIZE |
| INSERT | ROUND |
| GRAPH | DELETE |
| COPY | LIST |
| all "SET" words |

Table 2

In addition, a simple tool, CONDITIONS?, is provided for users to program their own music processing routines based on the system filter. CONDITIONS? is passed the address of the MIDI data as a parameter and
returns a flag indicating whether or not the filtering conditions were met.

Two means are provided for viewing filtered phrases: PAGE and GRAPH. These are, respectively, the alphanumerical and graphic representations of the musical data. With PAGE, all MIDI data is displayed along with the relevant data of duration, articulation, zero number and so on; the data that fall within the restrictions of the current scope are printed in reverse video (figure 2). With GRAPH, only the filtered data are shown (figure 3).

![Graph](image)

**Figure 2**

![Graph](image)

**Figure 3**
METRICAL POSITION

Table 1 above indicated that filtering decisions made on the basis of a note's position within a user specified rhythmic pattern were possible. This is an extension of editing procedures into the realms of implied characteristics. Because the rhythmic pattern constitutes a true musical grouping, this is the first step toward true context-sensitive editing, even though the data obtained (i.e. the note's position within a metrical structure) are derived implicitly.

The rhythmic pattern used may be very long in order to facilitate structural as well as local editing operations. For example a length of 64 measures of 4/4 time may be chosen as the underlying metre, with editing operations constrained to notes falling within bars 18 to 60. Conversely, a single bar of 4/4 may be set as the underlying metre, with operations constrained to the second and fourth beats of the bar.

By combining such operations at different temporal scales, it is possible to build large, rhythmically complex structures from simple elements (the author is currently using this feature to explore rhythmic aspects of Joseph Schillinger’s pre-computer algorithmic composition system). At present, all operations involving the use of the metrical pattern must confine themselves to a constant metre throughout the section of the score being edited. A clear need does exist, however, for this operation to be re-directed through a metre map of the score in order to accommodate changing metres. A graphic editor is provided for the purpose of editing the rhythmic pattern (figure 4).

![Metrical Position Diagram](image)

Current position: 2:04


(figure 4)
INTERVAL

All of the types of data listed above were either intrinsic (e.g. pitch, volume, etc) or implied (e.g. position in bar, note number). In order to extend the range of filtering conditions to include contextual attributes as well, the word INTERVALS (interval fetch) was added. This word, when passed the address of MIDI data as a parameter, returns the parameter interval between the data and the following note. While quite crude in terms of context sensitivity, this extension was adequate to allow editing based on the recognition of sudden changes of pitch register within a work (i.e. a large interval jump) or lack of such changes (i.e. a string of repeated notes). It also allowed users to experiment with interval-oriented composition (as in the style of later Stravinsky) where, for example, all diads constituting a rising minor sixth can be accented, or all groups moving by semitones can be made legato.

INTERVALS was implemented for pitch, velocity, and duration characteristics and placed within the existing filter structure to allow its combination with any of the methods of testing listed on the middle and right hand side of table 1. While its sensitivity is limited to the immediate context of adjacent notes, these is the first step to true context-sensitivity and has been very useful in extracting musically relevant features from algorithmically generated structures. With this system it is very easy, for example, to extract all repeated notes (pitch interval is equal to a constant) or pitch is equal to the next note). In addition, the implementation of INTERVALS for duration returns a relative rather than an absolute difference; for example, the duration interval between an eighth note and a quarter note would be .5 (their relative difference), and not 24, their absolute difference at 48 ticks per quarter note. The implications of this are far reaching; it allows the development of rhythmic motifs independent of their position relative to a metrical scheme and invariant with respect to time scale. Diminutions and prolongations are equally recognizable.

Also functioning in the immediate context are two of the functions listed on the extreme right hand side of table 1. These constitute the ability to compare any intrinsic or implied characteristic of a note with that of its immediate neighbour forward or backward (the next note or the previous note). A simple but extremely powerful extension of this was the ability to compare these characteristics to the next or previous tagged note (i.e. a note that has been marked by earlier filtering operations). This allows the establishment of relationships beyond the immediate context; it makes possible, for example, a selection based on the pitch of the current note, wherever it is in the bar, to the pitch of the next note that falls on a downbeat, or to the previous note that was higher than middle C and louder.
than MIDI velocity 96. In fact, since the tag markings are the logical sum of previous filtering operations, this effectively extends MIDIPOURTH's context sensitivity to any temporal limits desired, allowing relationships between distant sections of the composition to be established and acted upon.

THE SEARCH PATTERN

The most recent software device to be added to the MIDIPOURTH context sensitivity toolbox has been the SEARCH PATTERN, a list of up to 128 elements which may be used to define a musical pattern within the score. Like most MIDIPOURTH operations, the test for pattern identity may be directed to any musical parameter; this allows the search pattern to be used to specify such musically desirable groupings as:

- a pitch series (the original intention)
- a rhythmic motif (independent of its position within the move)
- a gradation of dynamics (i.e., a crescendo or diminuendo)
- a graded or patterned series of any other parameter (articulation, controller values, etc)

A very simple editor is provided for modifying the search pattern (figure 5).

![Search String Editor](image)

In order to facilitate its use in contexts other than pitch and rhythm (where a pattern may be precisely defined), the search pattern may be
generalized into a series of values representing, not precise intervals, but simply movements 'up' or 'down' from the previous value. This makes the recognition of dynamic patterns such as crescendi and diminuendi, for example, much less sensitive to local details when searching for membership in long patterns. It also makes possible the recognition of diatonically transposed pitch patterns. For example, it has been found that for pitch patterns longer than two or three notes this method is quite accurate in recognizing only legitimate members after a diatonic transposition.

FUTURE DIRECTIONS AND CONCLUSION

The directions in which such systems should proceed in the future include: more sophisticated forms of context grouping - for example, harmonic groupings, which are not adequately dealt with in the MIDIPOWTH system. Schenker-style harmonic and melodic prolongations are likewise true musical groupings that potentially provides a great deal of information about the structure of the work. More sophisticated pattern matching, of the type found in many text processing programs, is a clear extension. For example, wild cards where the number of elements is variable would be very useful musically. Another fairly simple but musically useful extension would be the ability to specify diatonic intervals - e.g. where '2' means either a major or minor second, depending on position within the scale.

Most of the musical relationships described in this paper have involved only two (or at most four) notes, although under some circumstances those notes might be widely separated in the composition. A very desirable addition would be the ability to make decisions based on large-scale contextual characteristics of a group of notes, such as the average pitch range (register) in the current phrase, or the long term trends in dynamics, ignoring local variations, that might indicate membership within a crescendo or diminuendo. Some provision has been made for this in the MIDIPOWTH system with the ability to compare the current note with a user defined formula (see table 1). This formula can access the musical data through well defined system constructs and act upon it in these ways.

While there is clearly still a great deal to be done, the concepts implemented in MIDIPOWTH constitute a true beginning and the first few steps in the direction of greater musical intelligence and power in the realm of computer-assisted composition. Whatever directions future systems may take, it is clear that the ideal is a sort of pattern and context recognition approaching the subtlety and sophistication of the
human mind. This, of course, brings us into the realm of artificial intelligence, and into contact with the long-standing debates regarding the feasibility of its objectives.

ACKNOWLEDGEMENTS

The development of the MIDIFORTH system has been financially assisted by the Ontario Arts Council and the Canada Council. Special thanks are due to John Free for certain ideas concerning the data structures used to represent music in MIDIFORTH.

BIBLIOGRAPHY


