Random Access to the Time Domain in the ANFLE language

Chris Jordan
Hybrid Technology Ltd
Cambridge
Great Britain

ABSTRACT

ANFLE is a powerful and versatile music programming language and environment for use on microcomputers. The language 'toolbox' includes a wide range of commands and control functions for advanced music applications, tailored to an interactive textual music notation (as more traditional musical forms). Higher-level components of the environment cater for specific user requirements such as instrument design, real-time performance control, and system control. The low-level software interfaces provide utilities access to a variety of musical input and output devices, including non-specific connections. Currently, the most popular use of ANFLE is in the World Music System for the BBC Microcomputer, including the Music 5000 Quintet.

ANFLE is a procedural language with a 'word' structure like that of LISP - the user program is a hierarchy of words, each defined in terms of pre-existing user words and pre-defined system words. Words can be created and edited individually, interactively, and in a variety of forms, always including the traditional textual program form.

The user program runs in real time under the control of a time manager that event and sorts events from all concurrent processes before playing. This allows remote access to the time domain over a range limited by system load and memory capacity, so that as well as the usual positive time intervals between events, negative time intervals can be accommodated. Time, processes and events are free to generate even sequences in non-linear orders that better suit the algorithms of the program or score. The major reward is simpler programming of a variety of overlapping musical structures, both large and small, with improved temporal control within, and, in particular, ANFLE's textual music notation.

Simple applications of the random time access facility are described, along with selected features of the ANFLE textual music notation.

NOTATION ELEMENTS

ANFLE music notation has words (symbols) for four basic types of musical event:

- **NOTATION ELEMENTS**

<table>
<thead>
<tr>
<th>Word</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>note of different pitches</td>
</tr>
<tr>
<td>rest</td>
<td>rest (silent note)</td>
</tr>
<tr>
<td>key</td>
<td>hold (continue last event)</td>
</tr>
</tbody>
</table>

  Other words set the parameters of events, the most important being 'time', which sets the length in time units. In scoring, a basic 'time' is added to words to which times have been added: all examples of this are achieved by extension of the 'word' held. For example:

  ```
  note time:
  ```

  On playing, X (also expands into a 'gave off' event followed by a 'variations' time interval: ' (repeat) expands into a 'give off' and duration/- into just a duration. In each case, one length of the duration is that set by the previous 'time', in this case (2) units.

  Hence, this single music event sequence triggers into a sound event sequence of just 'gave on' and 'give off' events with varying time intervals.

  The full set of ANFLE musical words is described in the Music 5000 User Guide (1986).

EG: EGG

**ENJOYED**

If a negative length setting is in force, the duration of each music event enters the 'time' process: each following music event, as the event will play before rather than after. Hence, the example could be reordered backwards as follows:

```

The resulting concert is the same, but there is a net backward movement in time rather than a forwards one, so subsequent music events will be displaced.

A-33

ICMC 86 Proceedings
The back-hold word, \(<\), has the effect opposite to that of the hold word, \(>\). It has the effect of a hold with the current length negated, as, for example, the sequence

\[
\begin{align*}
\text{ABC} & \quad \text{DEF} \\
\end{align*}
\]

has a zero net result.

One application is in the programming of the 'overlapping pick-up' musical structure - a short pattern of notes that leads into a section of a piece; for example a verse of a song. Thus can be a problem because the pick-up is temporally part of the previous section, yet must be functionally attached to its new section if this is to be called-up as a procedure in different contexts.

The negative duration is the pick-up is included in the 'rightful function' placed by allowing back-spacing from the section start to the pick-up start. Using back-holds, one is quite clear, for example:

\[
\begin{align*}
\text{ABC} & \quad \text{DEF} \\
\end{align*}
\]

The pick-up is quite free to overlay events at the tail end of the preceding section, generated by the same or another process.

**Chords**

Round brackets are used to denote chords, in which subsequent musical events of a group play alongside the first on successive voices. The subsequent events appear as a chorded group after the first 'main' event, for example:

\[
\begin{align*}
\text{C} \quad \text{(C) D} \\
\end{align*}
\]

In this simple example, the C, G and D in the first group play together as a major chord. In the third group, one voice starts a new note while others are held, and the final chord of rests finishes all voices.

For various reasons not discussed here, it is desirable to have a syntax like this in which additional voices are added after an unchanged main note. The availability of negative lengths allows this to be implemented by simple definitions of \(0\) and \(-1\), with no need for program look-ahead or retention of the last duration for possible retraction. The unspecified actions of the bracket words are as follows:

| word action comment                          | n, \  \ move back to main event start  \\
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n, \ restore net duration of main event</td>
<td>n, \ restore original 'G' setting</td>
</tr>
</tbody>
</table>

where

\[
\begin{align*}
\text{v} = \text{value of the main event} \\
\text{N} = \text{total duration inside brackets} \\
\text{Normal, } n = \text{since the length of bracketed event is 0.} \\
\end{align*}
\]

Additional actions (not shown) control voice selection.

**Pick-up Chords**

The bipolar duration implementation of chords allows more advanced chord-like groups to be scored very simply.

The programmer may explicitly use \(\langle\) to pause the brackets to get a non-zero length so that there is a time interval between bracketed events. This gives broken/chopped chord.

<table>
<thead>
<tr>
<th>normal chords</th>
<th>broken/chopped chord</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice (1) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (2) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (3) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (4) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
</tbody>
</table>

The brackets always that the net duration is always that of the main note. If the total length inside the brackets equals that of the main event, then a zero negative duration is instead the 'drum' overplay the next main event:

<table>
<thead>
<tr>
<th>normal chords</th>
<th>broken/chopped chord</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice (1) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (2) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (3) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
</tbody>
</table>

Alternatively, the complicating \(\langle\) nature can itself be negative so that the drum anticipates the main event:

<table>
<thead>
<tr>
<th>normal chords</th>
<th>broken/chopped chord</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice (1) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (2) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
<tr>
<td>Voice (3) =</td>
<td>(\text{CDEFG} -)</td>
</tr>
</tbody>
</table>

**Echo**

Echo simulated on successive voices is very similar to broken chords from the point of view of time domain aspects. The differences lie in the events that are directed to successive voices with successive time displacement of copies of the main event, rather than additional scored events. The word 'echo' is used to denote scalar effects.

The duration employs an event vector to eliminate duplicates of each event and translate time in time using / and 0. The interval between echoes includes negative + and the number of echoes can be specified.

**ICMC 16 Proceedings**

A-34
A particularly illustrative application of random access is simple control over articulation via the gate signal. Note-separating gaps of fixed length (non-legato or 'legato') are achieved by each note corresponding to a specified gate value and terminating its predecessor. Specfic style is achieved by the note or hit reangling forward to terminate itself a fixed time in the future.

Both these reaching actions have a zero set change on the time position, so can simply be added to the standard interpretation of note events along the event vector. The articulation actions can themselves be expressed in terms of simple music events - rest, hold and back-hold. This is more obvious when the articulations are thought of as chords added between notes:

\text{gsp (non-logical)}
\begin{align*}
\text{sequence} & \quad \text{start} \\
\text{event} & \quad \text{while} & \quad \text{while} & \quad \text{while} \\
\text{articulation} & \quad 12, x & \quad 12, x & \quad 12, x \\
\end{align*}

An interesting observation is that the articulation action is a simple form of 'gateoff' action one that expands each music event into a sequence of music events with derived parameters, normally used for more advanced compositional processes.

The simplified definition of the 'gap' action is as follows:

\text{ACT} & \quad \% start vector sequence
\text{\texttt{\%}} & \quad \% add leading rest
\text{\texttt{\%}} & \quad \% perform default note action
\text{\texttt{\%}} & \quad \% end vector sequence

Notice that the second line is identical in essence to the over-hanging pickup store - each 'operating gap' can be considered a rest that overhangs from the following note.

The staccato action is:

\text{ACT} & \quad \% start vector sequence
\text{\texttt{\%}} & \quad \% add trailing rest
\text{\texttt{\%}} & \quad \% perform default note action
\text{\texttt{\%}} & \quad \% end vector sequence

An important point to note here is that it is the amount of the note that adapts the fixed length of the rest, and the silent 'rest' portion is the variable-length remainder - the interval between the rest's 'gateoff' event and the next note's 'gate on'. The duration of the rest is negative, serving to return the time position to the start of the note.

The music event sequence on the second line of the definition is not only the most-impact representation (no back-rest is available) of the function, but also illustrates the communality in structure of the two actions. In practice, both these actions are performed by a single word 'gsp' targeted in a special case (when the length setting is a simple, positive or negative numeric argument).

Since durations can pass over existing events, there is nothing to prevent a 640 or static note interval being longer than the note themselves. Under these conditions, each note reaches beyond its immediate predecessor with successor to modify events further removed in the time stream. Complexed interactions of note length and sequence arise, giving interesting articulation and parsing effects. There is clearly much scope for further experimentation along these lines, for example, the use of multiple actions controlling a range of parameters as the basis for musical sequence generation and transformation in advanced compositional processes. This is well within the capabilities of current implementations of the AMUS language.

\section*{References}

\end{document}