Feature Classification and Related Response in a Real-time Interactive Music System

Robert Rowe
M.I.T. Media Laboratory
Music & Cognition Group
20 Ames St. Cambridge MA 02139
rowe@media-lab.media.mit.edu

Cypher is a real-time interactive music system that has two halves: a listener and a player. The listener listens to and analyzes external musical input. The player uses various algorithmic techniques to produce new musical output. The two halves are each hierarchical, where higher levels correspond to longer spans of time. The listener classifies for the player features detected in the input and their behavior over time, and the player uses this information to produce music in response. Collections of relations can be saved and recalled during performance by a score orientation section which tracks human performance and executes state changes at predetermined points in the score.

Cypher is an interactive computer program for composing and performing music. The program has two parts : a listener and a player. The listener characterizes the performance input of a player, which could be a human performer, another computer program, or even *Cypher* itself. The player generates and plays musical material. Both the listener and player are hierarchical structures, in which higher hierarchical levels correspond to longer spans of time. Let us sketch the basic functioning of the program: attuned to some MIDI source, the listener examines each event as it arrives, and produces a representation of it by characterizing the features density, speed, loudness, register, duration, and chord. On this lowest level of analysis, the program asserts that all inputs occupy one point in a *featurespace* of possible input classifications. The dimensions of this conceptual space correspond to the features extracted: one point in the featurespace, for example, would be occupied by high, fast, loud, staccato, C major chords. Higher levels look at the behavior of these features over time. The listener continually analyzes incoming data, and sends messages to the player describing what it hears. The user's task in working with *Cypher* is to configure the ways in which the player will respond to those messages. The player has a number of methods of reponse, such as playing sequences, or initiating compositional algorithms. The most commonly used method generates output by applying transformations to the input. These transformations embody small, simple operations such as acceleration, inversion, delay, or transposition. Though any single operation produces a clearly and simply related change in the input, combinations of them result in more complicated music. Specific sets of operations are performed on any given input event according to connections made by the user between features and transformations. Establishing such a connection indicates that whenever a feature is extracted from the input, the transformation to which it is connected should be applied to the event, and the result of the operation sent to the synthesizers.

Let us look at the listener's analysis by beginning with the lowest level. The first-order analysis places each incoming MIDI event in a multi-dimensional *featurespace*, a space framed by the following perceptual categories : register, loudness, density, attack speed, duration, and chord quality. Using these dimensions, the analysis identifies every input event as occupying one point in the featurespace. The values assigned to these categories are calculated as a function of the MIDI data associated with the input. For example, the *loudness* dimension is decided by simply dividing the velocity field into loud and soft according to some threshold. The order 2 level analyzes the output of level 1 for patterns of regularity or irregularity. Level 2 maintains a history of recent featurespace indices. Each time it is called, the level 2 analyzer looks through the history of the eight most recent featurespace points, and decides whether each feature classified by the point has been behaving regularly or irregularly over that span. A feature which has remained the same over the majority of the history is found to be *regular*, and one which has changed more often than not is found to be *irregular*. As with the level 1 analysis, the regularity findings are continually passed over to the player; the user can then configure the player to respond in various ways to specific regularity findings.

The generation section of *Cypher* is designed to accomodate three algorithmic styles: 1) transformation of material coming from outside sources, 2) purely generative algorithmic processes, and 3) performance from a library of sequences. The most heavily used method involves the transformation of material, accomplished through the chaining of many small, straightforward modules. The action of these modules is cumulative: if more than one is used, they are applied serially, with the output of one operation being passed to the input of the next. Though the action of any module taken singly is simple and easy to follow, longer chains of transfomations can build up material that is quite complex, though deterministically derived from its inputs. Among the transformations already implemented are the following: acceleration, accentuation, arpeggiation, reversal, chord formation, deceleration, glissando generation, inversion, crescendo, diminuendo, transposition, etc. Level 2 generation processes are used to mutate the transformation modules; control variables relevant to the transformation method can be changed according to the nature of the input being reported by the listener and the current state of the variable. For example, the accelerando module can be made to accelerate its input more or less than it already does as a function of the speed feature found by the listener, and the accelerando rate currently active. The *Cypher* listener listens to a player. The player can be any stream of MIDI input, including the output of *Cypher*'s own player. Sending the output of the generation section to the input of the listener causes composition by introspection. Music is generated from a controlled feedback loop: the player produces some output, which is analyzed and characterized by the listener. The listener sends messages to the player about what it has heard (in other words, what the player just produced), and the player is instructed, through the connections with the analysis features, to transform its own previous output in some way.

The basic listening model in *Cypher* combines the output of many small, relatively simple agents, following low-level features such as register, density, and loudness, and the way these change over time. As we have seen, one important facet of their behavior in time is their regularity, and rate of change. Others include grouping, the way events are divided into larger structural units, and direction, whereby we attempt to see some uniformity in the way a feature is changing which might allow us to predict its future behavior. A simple example of finding direction would be noticing an accelerando, or crescendo. The performance of high-level tasks, such as chord and key identification, or beat tracking, arises from the communication and mutual influence of many agents on each other and on the higher-level agencies. These high-level tasks, in turn, inform and direct each other in performing their own computations, and in contributing to a description of the musical context as a whole.