A COMPUTER PROGRAM FOR ANALYZING COMPUTER MUSICIANS' PROBLEM-SOLVING

Richard D. Ashley
School of Music
Northwestern University
711 Elgin Road
Evanston, Illinois 60208 USA
(708) 491-5431
email : rdt@music.nwu.edu

ABSTRACT: This paper describes a computer program, written in Common Lisp, which seeks to provide a basis for the automated analysis of a computer musician's activity, with an eye to understanding the problem-solving methods employed. The particular activity under consideration here is that of young musicians working with a tutorial system for aural analysis. The system represents connections between musical structures in a network, which can be used for seeing the ways in which different events in a musical work are related to one another and analyzes student's actions in terms of "plans" which contain typical problem-solving behaviors.

With the considerable potentials of computer music systems come considerable complications. Among these are issues of good user interface design, driven by a desire to produce computer systems which are both effective (i.e., which get the job done) and efficient (are well-designed so that a user's task is not unnecessarily complicated). In order to produce systems with those characteristics, the nature of the tasks for which the system will be used needs to be well-understood, as well as the ways in which computer musicians make use of the resources of a system to get musical goals accomplished. Attempts have been made to carry out studies of computer users in other domains in the context of cognitive theories (Card, Moran, & Newell 1983) and to understand problem-solving behaviors with the aid of computer analysis programs (Ericsson & Simon 1984), but few if any attempts have been made to apply such ideas to musical undertakings (for one exception, see Lake 1977). We have implemented a computer program, written in Common Lisp, which seeks to provide a basis for the automated analysis of a musician's activity, with an eye to understanding the problem-solving methods employed in dealing with a computer music system.

We are examining the activities of young musicians working with a tutorial system for aural analysis (Ashley 1989). In this system, students learning is viewed as heuristically-guided discovery. The task with which the student is confronted—that of developing a rich internal model of the work at hand—is relatively ill-defined in terms of a discrete end-state and procedures by which the end-state might be achieved. The student must engage in a kind of problem-solving which involves defining the actual conditions under which the overall goal has been achieved, and specifying the means for attaining the goal. The actual path which a student takes in satisfying the goal will vary according to a number of factors, especially prior knowledge and experience with the style and structure of other pieces of music. The student comes to understand the structure of the work, and to learn the work thoroughly by ear, using only the piece in its aural form and a graphic interface. The activity on which we are currently focusing involves describing the sequence of events in the work. On the computer, the student's task is to reconstruct the piece of music by arranging "blocks" of music on a timeline, where they can be played in order when desired (see Fig. 1). This activity is designed to begin orienting the student to the whole piece.

The tutorial system collects protocols of student activities which can then be analyzed to further refine the system's tutorial aspects. Traditionally, data on problem-solving tasks have been focused on two main elements: external events such as "states" of a puzzle or game board, and verbal protocols. In this instance, we collect information on the keystrokes/mouse-clicks.

ICMC GLASGOW 1990 PROCEEDINGS

144
performed by the program user. Such data is not novel; some computer programs (such as the Common Lisp language) even provide primitives which provide such transcripts or "dribble files." Our system adds the element of timing of events. We thus have a "computers' eye view" of the student's work, where every user action with regard to the computer is recorded in detail; this corresponds to state information like that found in traditional game- or puzzle-solving tasks.

Once the data has been collected, it must be subjected to analysis, which is typically a very tedious and time-consuming procedure. Our protocol analysis system, called Lispro, has been designed to support, but not replace, the human analyst's role. Lispro is a forward-chaining production system (a kind of rule-based program architecture) written in Common Lisp and in KSM, a production system language (Ashley 1985). Its task is to perform a kind of rough first analysis of the student's performance data, which it does by breaking up the large sequence of actions into smaller problem-solving steps. It also shows the kinds of knowledge which a student might glean from the piece at any point in time. In this way the person studying the student's activities can spend more time on the harder parts of the analysis, since the routine aspects have already been covered. What is needed is a means of making at least an approximation to the meaning of the student's actions—to seeing what each keystroke or mouse-click means. This is one of the things a human interpreter of the data does; every action is accorded some kind of intentionality or goal, according to its context.

The kind of "discovery learning" at work here can be seen as having two main activities: hypothesis generation and hypothesis testing. We are now using seven primitive actions derived from the puzzle activity. These are:

Information-Collecting Actions
- seek-section: look for a certain kind of clue
- check-model: check the current model or some part of it
- compare-sections: compare two or more sections to one another

Model-Altering Actions
- add-to-model: add a segment to the model
- remove-from-model: remove a segment from the model
- reorder-model: change the ordering of segments within the model.

The rules in Lispro are of three main types, each gathered into a "rule set." The first set of rules is "perceptual" in nature, and gathers information for the rest of the system. This rule set examines the actual transcript of the student's activity and obtains the most important information from each action. The second set of rules, which generalizes from an individual move to one

ICMC GLASGOW (1990) PROCEEDINGS
of the seven primitive actions. For example, if an icon corresponding to a section of music has been clicked, Lispco determines this fact. This information is used by the second set of rules, which generalize, mapping a student action onto one of these seven primitives. These are then added to the student transcript, parsing the relatively large number of actions (about 200-300 in a typical 30 minute working session) into a series of large problem-solving “blocks.” These blocks can then be examined for finer detail by the Lispco’s user. The final set of rules attempts to take these sequences of blocks and determine the overall problem-solving “plans” or methods which the student is employing.

Currently, Lispco recognizes two main problem-solving plans in the puzzle task, and alerts the user of the presence of these. The first plan is straight left-to-right-with-backtracking, or “walliding.” A student working with this problem-solving method will try to locate blocks in the puzzle starting from the beginning of the piece and working through systematically to the end. Usually, this is signaled by listening to the piece from the beginning up to a certain point, locating the section which corresponds to the latest point in the music which has been heard, then returning to the front of the piece, listening a bit further, and continuing in this manner until the entire work has been reconstructed. This method, although usable, is not too efficient due to the wasted information which could be collected while searching for the correct section of the piece. It is not unusual to see students begin with the bulldozing strategy and then move to an “opportunistic” mode, in which each smaller section heard will be put into the puzzle as soon as possible, even if only on a provisional basis. When Lispco sees that the bulldozing strategy is not being employed, it proposes that a kind of “island-driving” process (Erman et al. 1980) is at work, where the student will extract any useful information from the section of music being heard (such as instrumentation, and ry to use it immediately. To this end, Lispco shows the user the kinds of information which are available to the student (stored as information about the contents of each section of music) so that detailed analysis may begin on the kinds of relationships: with which the student is dealing.

The goal of this project is to develop a set of normative problem-solving procedures, which can then be taught to the students implicitly, through the design of the curricular materials themselves, or explicitly, as procedures and strategies. This may be in the form of heuristics, but is likely to involve more work in designing and implementing a plan-based tutoring system, which would move not toward giving the student "correct" answers (difficult in some of these activities!) but toward moving the student flexibly and unobtrusively into activities and avenues of investigation which are likely to be beneficial to him.

References


ICMC GLASGOW (1990 PROCEEDINGS)