Cirrus: Inducing Subject Models From Protocol Data

Bernadette Kowalski and Kurt VanLehn
Departments of Computer Science and Psychology
Carnegie-Mellon University

INTRODUCTION

Verbal protocol data are collected by asking subjects to talk aloud as they solve a problem, and protocol analysis is the process of interpreting such data. Protocol analysis is used routinely by psychologists and other behavior scientists, and more recently, by knowledge engineers who wish to embed the knowledge of human experts in an expert system. However, protocol analysis is notoriously difficult and time comsuming.

Several systems have been developed to aid in protocol analysis. Waterman and Newell (1971, 1973) developed a system that could read the natural langauge of the protocol and produce a formal trace of it (a problem behavior graph). The system, however, did not produce an abstract model of the subject. Bhaskar and Simon (1977) avoided the natural language understanding problem by having a human coder read the protocol and formalize it. The Bhaskar and Simon program, however, was not designed to construct a model of the subject, but instead to test a specific model of thermodynamics problem solving that was built into it. Fisher's (1988) system provides facilities for assigning formal codes to sections of protocols, interrogating a database of codes/sections, performing path analyses, and inferring flow-chart-like models of the subject's behavior.

Cirrus is halfway between the theoretical neutrality of Fisher's program and the theoretical commitment of Bhaskar and Simon's program. It requires the user to provide an underdetermined model, namely, a problem space (Newell & Simon, 1972). A problem space consists mainly of a representation for states and a set of state-change operators. It is not a complete model of the subject's behavior because it does not contain information for selection of operators or goals. The job of Cirrus is to infer from the data the subject's selection strategies. A completely determined, subject-specific model thus consists of the problem space given to Cirrus by the user and the selection strategies constructed by Cirrus to maximize the fit with the subject's protocol data.

Cirrus requires a human to encode the verbal protocol. However, Cirrus can not accept all formal codes, but only those that designate actions (i.e., the application of a primitive state-change operator). Moreover, Cirrus requires that *all* the subject's actions be encoded. A protocol that consists of all and only the actions of the subject is called an *action protocol*. Typically, action protocols are collected by implementing the experimental task on a laboratory computer and saving the subject's keystrokes. A verbal protocol is usually taped at the same time because subjects often make revealing comments about goals, plans, rationales, difficulties, etc.

Cirrus and the human analyst have distinct jobs in analyzing a protocol. The human analyst uncovers the subject's problem space by drawing upon common sense and knowledge of the task domain (something that Cirrus has none of) in order to interpret the whole protocol, both actions and

non-action commentaries. Cirrus then tests the adequacy of the human's analysis and deepens it by trying to find selection strategies that will accurately postdict the subject's actions. If some sections of the action protocol are not adequately fit, then the human theorist can re-examine the protocol at those points, revise the problem space, and run Cirrus again. In short, Cirrus is a data analysis tool and not an automated protocol analyst.

This paper describes the current version of Cirrus. (An earlier version is reported in VanLehn and Garlick, 1987.) The first section describes the theory of problem solving that Cirrus assumes. Although the theory is quite standard and noncommittal, it is nonetheless the source of most of Cirrus' limitations as an analytical tool. The second section describes how Cirrus works. The third section discusses Cirrus' analysis of protocols from two different task domains. The last section discusses our plans for removing the current restrictions imposed by Cirrus' theory of problem solving.

ASSUMPTIONS ABOUT PROBLEM SOLVING

This section presents some assumptions about the representation and interpretation of procedural knowledge. All of the assumptions are fairly standard in the field, but the particular combination of assumptions used in Cirrus need to be made explicit. First, the basic problem solving architecture will be described, then the assumptions behind it will be discussed.

The basic cycle of problem solving is to (1) select a goal, (2) select an operator that is relevant to achieving that goal, (3) execute the operator, and (4) delete the goal. Some operators are primitive, in that executing them changes the state of the problem. Other operators are macro-operators, in that executing them causes new goals to be created. This basic cycle is initiated with a top level goal; it finishes when there are no goals left.

The knowledge representation consists of (1) a set of operators, (2) a strategy that determines which operator to choose for any given goal, (3) a strategy for determining which goal to choose, and (4) a state description vocabulary, which is a list of the attributes of states that are considered relevant for making strategic decisions. The user provides Cirrus with the set of operators, the operator selection strategy and the state description vocabulary. Cirrus infers the subject's goal selection strategy. Soon, Cirrus will also be able to infer the subject's operator selection strategy as well. This capability is a standard technique in machine learning (see, e.g., Langley and Ohlsson, 1984), and we anticipate no problems incorporating it in Cirrus. Thus, Cirrus will require only the problem space (i.e., operators and state description language) from the user, and will infer the rest of the subject model itself.

There are several tacit features of this problem solving architecture that distinguish it from others in the GPS/STRIPS class. First, it lacks GPS's commitment to difference reduction as the method for choosing operators. The operator selection strategy can, in principle, be any function of the current state, and not one that reduces the difference between the current state and the goal. Currently, Cirrus uses a simple representation for operator selection strategies. Each operator has a condition attached to it. After the architecture has found which operators are relevant to the current goal, it tests the conditions attached to the operators. If exactly one operator has a true condition, it is selected; otherwise an impasse occurs (Brown & VanLehn, 1980). Currently, Cirrus does not model subject's reactions to impasses.

In order to back up during problem solving, most GPS-style problem solvers have a state selection

phase that immediately precedes the goal selection phase. Cirrus does not, in part because backing up is not that common, and when it does occur, the subject often just starts the problem over (Newell & Simon, 1972). When starting over occurs, it is usually easy for a human analyst to spot, so the coder can merely given Cirrus a protocol that has two solutions of the same problem, the first of which happens to end in failure. Thus, Cirrus can "handle" the most common types of backup, albeit crudely.

Like GPS, Cirrus does not model learning during the course of problem solving.

Like GPS, Cirrus has very limited planning abilities. Subjects sometimes plan ahead by mentally simulating the next few actions. Cirrus can not model this. Nor can Cirrus model abstraction planning, such as that used by Newell and Simon's logic subjects. Nor can Cirrus model envisioning, wherein a subject mentally simulates the working of a physcial device. Cirrus can not do these things because it only represents one problem state. Usually, the current problem state represents current state of the real, physical world, although it can represent an imaginary world, as when the subject is doing mental mulitplication. Cirrus can not represent someone who is working in both a mental world (often as a way of developing a plan) and a real world. This means that the only kind of planning that Cirrus can represent is operator subgoaling, wherein one develops a stack of subgoals before finally arriving at an operator that can be executed directly (e.g., "In order to get from Pittsburgh to Montreal, I'll fly; but a precondition of flying is being at the airport, so I'll take a cab to the airport; but a precondition of taking a cab is ...").

However, Cirrus does have a flexible approach to subgoaling. It does not insist on a depth-first traversal of the goal hierarchy. Instead, it assumes people have a goal selection strategy. For instance, if taking a cab spawns the subgoals of (1) being at a cab and (2) having enough money to take a cab, then the goal selection strategy would have to decide which of these new goals to work on. Indeed, it could even decide to work on an older goal, such as obtaining money for airfare. Thus, Cirrus can model subjects who jump around in the goal hierarchy. This has turned out, somewhat surprisingly, to be necessary even for modeling simple skills, such as subtraction (VanLehn & Ball, 1987).

The goal selection strategy of Cirrus is represented as a set of conditional preferences of the form "Prefer <goal-1> over <goal-2> when <condition>." For instance, in subtraction, it is important to execute borrowing actions that effect the top digit of a column before one answers the column. To represent this, one can use preferences such as "Prefer (Add10 ?C1) over (Diff ?C2) when C1=C2." This says that when both the goal of adding ten to the top of a column and the goal of taking the difference in a column occur, and they refer to the same column, then one should do the addition of ten first. Such preferences are used in many contemporary problem solvers, such as Soar (Laird, Newell, & Rosenbloom, 1987) and Prodigy (Minton et al., 1987).

Cirrus has been described as problem solver. However, that is only a small fraction of its job. Its main task is to find the goal selection strategy of the subject (and, soon, the operator selection strategies as well). When those are found, the complete model of the subject is executed by the problem solver in order to measure its fit to the protocol.

THE ANALYSIS METHOD

This section describes the techniques used by Cirrus to infer the goal selection strategy. Note that, in contrast to the preceding section, these processes are *not* claimed to be psychological processes. Neither the subject nor the human analyst does anything remotely similar to what is described in this

section.

The first step in the analysis of an action protocol is to parse the protocol using the operator set provided by the user. The algorithm used is a modification of a standard top-down algorithm for parsing with context-free grammars. The result, in the ideal case, is a single tree for each problem. The tree represents the goal-subgoal decomposition that the subject took on that problem. The leaves of the tree are the primitive operator applications that constitute the subject's actions. In the less-than-ideal case, parsing a problem produces more than one tree. Cirrus asks the user to choose. In the worst case, some problems can not be parsed at all. This indicates an inadequacy in the set of operators which the user must correct.

The second step in the analysis is to convert each goal-subgoal tree into the trace that the problem solver would produce if it had generated this tree. The trace contains each operator selection event and each goal selection event that the problem solver would have had to perform. Cirrus generates this trace by traversing the tree, and outputing the appropriate selection events each time it visits a goal in the tree. Unfortunately, the trace is not always uniquely determined. Sometimes the tree can be traversed in hundreds of ways while still remaining faithful to the chronology of the action protocol. Cirrus currently uses heuristics to guide the tree traversal. The heuristics are based on the assumption that solvers tend to minimize their working memory load. The heuristics guide the tree traversal along a path that minimizes the number of active goals. It is not yet clear whether this heuristic will always be a good one or how to detect when it has led the analysis off on a garden path. This would be a good place for Cirrus to have access to the non-action parts of the verbal protocol, because some of the subject's comments might indicate in which order the goals are considered.

The next step in the analysis is to convert the goal selection events into preferences. This occurs in two phases. First, preferences are constructed without conditions attached to them. Such preferences can be either *consistent* with a goal selection event, or *inconsistent*, or *irrelevant*. Thus, if the preference is "Prefer A over B", and the event is that goal A is chosen when both A and B exist, then the preference is consistent with the event. The preference would be inconsistent with the event if B were chosen instead of A. The preference is relevant only to events where both A and B exist. Preferences are constructed only if they are consistent with at least one event.

The second phase builds conditions for preferences. If a preference is consistent with some events and inconsistent with others, then a condition is induced that is true of all the consistent events and false of all the inconsistent ones. The condition is a boolean combination of attribute-value pairs drawn from the state description vocabulary. (This is the main reason for having a state description vocubulary.) The ID3 algorithm is used to perform condition induction (Quinlan, 1983). There are some biases built into ID3 as well as all other condition induction algorithms. We have experimented with several other induction algorithms: the Chi-squared variant of ID3 (Quinlan, 1986), the standard version space algorithm (Mitchell, 1982), a noise-handling version space algorithm (Mitchell, 1978), and some algorithms of our own. Our experience is that the bias inherent in the condition inducing algorithm is not as important as the bias inherent in the state description vocabulary. An inadequate vocabulary will hurt them all, and a good vocabulary allows all of them to succeed, more or less. We use ID3 because it seems best at handling noisy data.

The last step in the analysis is to test the model by executing it and comparing its "protocol" with

the subject's. This step is purely for the user's benefit, because the current version of Cirrus can not go back and reconsider the heuristically determined choices it made during analysis. The user is responsible for feeding the results of the analysis back into the Cirrus, typically via changes in the state description vocabulary.

The original version of Cirrus (VanLehn & Garlick, 1987) used a completely different representation for goal selection stategies and a completely different method for inferring it. Kowalski and VanLehn (1988) discuss its shortcomings and the experiments that led to the current version of Cirrus.

RESULTS

Cirrus has been extensively tested on protocols collected from eight grade-school students solving multi-column subtraction problems (Kowalski & VanLehn, 1988). Preliminary testing has also been done on two protocols collected via the use of Sketch, a tutoring system that trains students to graph transcendental equations (Trowbridge, Larkin & Sheftic, 1987). This section discusses our experiences using Cirrus to analyze these data.

The subtraction protocols were collected as part of a study reported earlier (VanLehn & Ball, 1987), so the subjects, methods and results will not be detailed here. The chief finding is that eight subjects in a biased sample of 26 third-graders executed subtraction procedures in non-standard orders. For instance, some students would do all the borrowing in a problem first, moving from right to left, then fill in the answers to the columns, moving from left to right. A second finding is that students frequently change their execution ordering, often in the middle of solving a problem. This makes it seem likely that they are using a non-standard goal selection strategy that is conditional on the characteristics of the problem state, rather than having learned a non-standard algorithm for solving subtraction problems. However, when these protocols were analyzed by hand, we were only partially successful at finding goal selection strategies that would explain the subjects behavior. Up to one-third of the choice points in some protocols were left undetermined (VanLehn & Ball, 1987). The tedium of trying to improve this fit was a major motivation for the Cirrus project.

When Cirrus analyzes the action protocols of the eight non-standard order subjects, the subject models obtained are nearly perfect. Except for one ambiguous choice point in the protocol of one subject, the models infered by Cirrus exactly match the protocols. Not only does Cirrus do the job it was designed to do, it outperforms human analysts in its ability to match the students' protocols.

However, perfect matching of protocols is to be expected from Cirrus. The induction techniques used in Cirrus are powerful enough to analyze any goal selection strategy, provided that the user has the patience to keep augmenting the state description vocabulary until Cirrus finds appropriate conditions for all the preferences. Thus, the appropriate measure of Cirrus' performance is whether the preferences it finds make sense as goal selection strategies. This is not an easy evaluation to make objectively, so we can only present our experiences. In general, when Cirrus builds a small condition (3 to 5 attributes), our intuition agrees with its condition; the student really does seem to have that preference. However, when Cirrus builds a large, deeply nested condition (7 to 9 predicates), our intuition is that the subject probably has no preference (i.e., they choose whimsically between these two types of goals) or has a define preference but occasionally slips in applying it. However, during the early stages of using Cirrus on these protocols, most of the large, uninterpretable conditions turned out to be due to inadequacies in the state

description vocabulary.

The Sketch data were collected directly from the user interface of a tutoring system. Preliminary analysis of the Sketch protocols showed that Cirrus is a feasible system to use to analyze these in greater depth. The protocols share a characteristic with the subtraction protocols in that subjects tend to alter subtask ordering depending on the problem context. Cirrus has done an adequate job of characterizing the problem features that lead to the reordering of subtasks, although a theorist has not yet spent time in the loop with Cirrus refining the state description vocabulary to fully capture the patterns in the data.

DISCUSSION AND FURTHER WORK

Cirrus has potential both as a data analysis tool for scientists and as the student modelling component of an intelligent tutoring system (VanLehn, 1988). Indeed, Cirrus grew out earlier two student modellers for the subtraction task, Debuggy (Burton, 1982) and ACM (Langley & Ohlsson, 1984). Like Cirrus, these systems analyze data off-line, because they are too slow for real-time use in an intelligent tutoring system. A more important limitation is that all three systems, Cirrus, ACM and Debuggy (but not IDebuggy, a variant of Debuggy), expect data from all the problems to be available initially, whereas real-time student modelling requires taking in problem solutions as they are generated, and revising the student model incrementally. Incremental versions of all the analysis algorithms used by Cirrus exist in the machine-learning literature, but we have not yet tried to convert them because we are, at present, more interested in developing Cirrus as a tool for scientists.

The most important direction for improving Cirrus is to relax the assumptions about problem solving that are built into it. We would like to use Cirrus to analyze protocols of college students solving physics problems, which were graciously given to us by M.T.H. Chi. Almost all the work on physics to date has used coarse-grained protocol analyses, such as constrasting the order in which novices and experts write equations. This misses some interesting behavior, such as planning and envisioning, which is revealed in some of the subjects' verbal commentary. As mentioned earlier, Cirrus' assumptions do not allow for "dual world" problem solving, so this kind of behavior can not be modelled. Significant extension to Cirrus will be required.

The current version of Cirrus is definitely useful, although it is still a research prototype rather than robust, distributable software. Nonetheless, it demonstrates that human protocol analysis can be aided by computer-based tools. It has taken longer to develop these tools than it took to develop statistical packages because the underlying machine learning technology was developed only recently. Nonetheless, the way seems clear for further development of protocol analysis tools, which should allow much easier interpretation of a particularly rich and revealing window on human cognition.

REFERENCES

Bhaskar, R. & Simon, H. A. (1977). Problem solving in a semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, *1*, 193-215.

Brown, J. S. & VanLehn, K. (1980). Repair Theory: A generative theory of bugs in procedural skills. *Cognitive Science*, *4*, 379-426.

Burton, R. B. (1982). Diagnosing bugs in a simple procedural skill. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic. 157-183.

Fisher, C. (1988). Advancing the study of programming with computer-aided protocol analysis. In G. Olson, E. Soloway & S. Sheppard (Ed.), *Empirical Studies of Programmers*. Norwood, NJ: Ablex.

Kowalski, B. & VanLehn, K. (1988). *Induction of partial orders beats classification: Improvements to the Cirrus protocol analysis system* (Tech. Rep. PCG-15). Dept. of Psychology, Carnegie-Mellon University.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An Architecture for General Intelligence. *Artificial Intelligence*, *33*, 1-64.

Langley, P. & Ohlsson, S. (1984). Automated cognitive modeling. In *Proceedings of AAAI-84*. Los Altos, CA.: Morgan Kaufman,

Minton, S., Carbonell, J.G., Etzioni, O., Knoblock, C. & Kuokka, D.R. (1987). Acquiring effective search control rules: Explanation-based learning in the Prodigy sytem. In P. Langley (Ed.), *Proceedings of the Fourth International Workshop on Machine Learning*. Los Altos, CA: Morgan Kaufman.

Mitchell, T.M. (1978). *Version spaces: An approach to concept learning* (Tech. Rep. STAN-CS-78-711). Computer Science Department, Stanford University.

Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203-226.

Newell, A. & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Quinlan, J. R. (1983). Inductive inference as a tool for the construction of high-performance programs. In R. S. Michalski, T. M. Mitchell & J. Carbonell (Eds.), *Machine Learning*. Palo Alto, CA: Tioga.

Quinlan, J. R. (1986). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Ed.), *Machine Learning: An Artificial Intelligence Approach. Volume II*. Los Altos, CA: Morgan Kaufman.

Trowbridge, D., Larkin, J. & Scheftic, C. (1987). A computer-based tutor forgraphing equations. In *Proceedings of National Education Computer Conference*. Eugene, OR: ICCE,

VanLehn, K. (1988). Student modelling. In M.C. Polson & J.J. Richardson (Ed.), *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Erlbaum.

VanLehn, K. & Ball, W. (1987). *Flexible execution of cognitive procedures* (Technical Report PCG-5). Carnegie-Mellon University, Dept. of Psychology.

VanLehn, K. & Garlick, S. (1987). Cirrus: an automated protocol analysis tool. In Langley, P. (Ed.), *Proceedings of the fourth Machine Learning Workshop*. Los Altos, CA: Morgan-Kaufman.

Waterman, D.A. & Newell, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial Intelligence*, *2*, 285-318.

Waterman, D.A. & Newell, A. (1973). *PAS-II: An interactive task-free version of an automatic protocol analysis system* (Tech. Rep.). Department of Computer Science, Carnegie-Mellon University.