Machine Learning 2: 247–276, 1987

Theory Change via View Application in Instructionless Learning

JEFF SHRAGER (SHRAGER@XEROX.COM)

Intelligent Systems Laboratory, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304, U.S.A.

(Received: August 15, 1986)

(Revised: August 12, 1987)

Keywords: Cognitive modelling, induction, conceptual combination, theory formation, categorization, analogy, device model learning.

Abstract. IE is a model of learning by experimentation in the domain of complex devices. In this paper we describe *view application*, the principal component of that model. This mechanism combines abstract knowledge structures into the learner's theory of the device. View application organizes complex changes in the learner's theory, thus ensuring that the space of theories is searched rapidly and that only coherent theories are tried. We evaluate the mechanism along three dimensions - its psychological validity, its generality, and its ability to constrain search. We also compare view application to other knowledge-rich learning techniques.

1. Introduction

Humans are often confronted with complex devices in which the inputs and outputs are mediated by hidden internal states, and for which there is little or no instructional material. Examples of such devices include automatic teller machines, computer operating systems, automobile features, and microwave ovens. Faced with the problem of learning to operate such systems, one can sometimes "figure out" the device by exploration and experimentation. We call this the task of *instructionless learning*. Our overall goal is to construct a theory of learning in such information-limited situations.

There is a great deal of psychological research that, in principle, could provide insights into the nature of instructionless learning. This includes research on geometric induction (Bruner, Goodnow, & Austin, 1956; Hunt, 1962); the induction of mathematical functions (e.g., Huesmann & Cheng, 1973); sequence extrapolation (e.g., Kotovsky & Simon, 1979); and the broad literature in "concept identification" (e.g., Bower & Trabasso, 1963).

However, these domains contain no hidden entities or values, and the subject generally knows the structure of the stimulus space beforehand (e.g., simple geometric figures of various colors). Furthermore, these paradigms generally prescribe the actions available to the subject (e.g., indicating that an exemplar is or is not a member of the "correct" set). This prevents the subject from changing his data-gathering strategy during the task, and from using a favored or novel strategy. In contrast, in instructionless learning situations the device to be understood often exhibits complex behaviors (e.g., mode-dependent and programmable behavior), and the learner does not know the relevant features a priori. Also, there are few constraints on the learner's actions; for example, he can try out different data-gathering strategies to discover those best suited to the problem at hand.

The literature on scientific reasoning is more relevant to our goal. Instructionless learning problems are analogous to scientific induction tasks, in which the scientist attempts to understand a complex system containing hidden aspects. Unfortunately, scientists make difficult subjects for a variety of reasons: it is difficult for them to give verbal protocols, they tend to be very familiar with their problem domains, and their behavior is entangled with their scientific culture and training. Therefore, theories of scientific induction are generally inferred from historical facts and notes (e.g., Darden, 1976; Langley, Simon, Bradshaw, & Zytkow, 1987; Zytkow & Simon, 1986) or by placing students in model situations (e.g., Mynatt, Doherty, & Tweney, 1977). We hope that an improved understanding of instructionless learning will lead us to better underpinnings for a psychology of scientific reasoning.

Elsewhere (Shrager, 1985; Shrager & Klahr, 1983, 1986) we have reported a psychological experiment in which Carnegie-Mellon University undergraduates were asked to "figure out" a complex programmable toy tank without instruction or advice. The subjects were permitted to interact with the toy in virtually any (non-destructive) way. This experiment is more appropriate to our goals than the simpler studies that generally populate the psychological literature on learning. We developed a theory of instructionless learning in this situation, constructing a cognitive simulation of our subjects' behavior in order to refine and validate the theory.

This paper focuses on *view application*, the principal component in our computational model of instructionless learning. This mechanism uses preexisting abstract schemas to restructure the learner's knowledge of the device. Although view application is only one of several mechanisms in the model, it plays the largest and most interesting role. (Other learning mechanisms include structural analogy, causal reformulation, and simple generalization.) An important assertion of our theory is that a simple theory reformulation mechanism can succeed in complex domains, provided

it can draw upon prior knowledge and provided it is embedded in a learning system that can interact with the target and so uncover errors. View application is such a mechanism.

We begin by summarizing our empirical results on instructionless learning, which are reported more thoroughly in Shrager and Klahr (1986). We then summarize our theory of instructionless learning and the associated computational model, focusing on the role played by view application in theory change. We next describe the view application mechanism in detail, giving an example of its functioning. After this, we evaluate this mechanism in terms of its psychological adequacy, its ability to limit search, and its generality. We conclude with remarks on the relation of view application to other learning theories, along with some directions for future research.

2. Psychological observations on instructionless learning

In our experimental study of instructionless learning (Shrager & Klahr, 1986), we asked seven undergraduates to independently "figure out" the BigTrak, a programmable battery-operated tank-like toy vehicle made by the Milton Bradley toy corporation. The BigTrak can be programmed by pressing a sequence of buttons on a keypad attached to its back. For instance, pressing the buttons **CLR FORWARD 3 RIGHT 1 5 HOLD 5 0 BACKWARD 1 GO** will make the BigTrak move forward 3 feet, turn right 90 degrees, pause for 5 seconds, then back up one foot. Pressing **CLR** resets the BigTrak's program memory. If **CLR** is not pressed, new steps are appended to ones already in memory. Turns operate in units analogous to minutes on the face of a clock: 15 units represents 90 degrees. **HOLD** operates in tenths of seconds. There is also a simple looping construct, as well as ways to check and edit the program.

Several features make instructionless learning in this domain interesting. The BigTrak is a complex electronic and electromechanical toy. Most subjects develop an electromechanical theory of the workings of the device instead of (or in addition to) a description of its grammar. Also, the BigTrak produces complex behavior when buttons are pressed, instead of simply giving a Yes/No response. Learners use the actions of the device to guide them in theory formation. In the instructionless learning paradigm, the subject is permitted to interact with the device in virtually any (non-destructive) way. Since the cost of experimentation with the BigTrak is low, there is nothing to discourage many such interactions.

In 30 to 45 minutes, each subject developed a theory of the BigTrak that was correct in most aspects. During the session, subjects tried to validate the correctness of each theory change by performing "experiments" that would confirm the theory. They also performed "exercises," which

tested the learner's overall knowledge of the device; and "explorations," which were used to refine incomplete hypotheses. The protocols contained between 33 and 68 interactions with the BigTrak (*mean* = 52, *standard deviation* = 12.9, $n = 7$). The mean time between interactions (less BigTrak run time) was 27.7 seconds (*standard deviation* = 5.7). The interactions classified as exercises generally increased in length over the session.

All subjects seemed to interpret the instruction to "figure out" the BigTrak as a request to assign a particular role to each physical aspect of the device. At some level, all the subjects formulated a theory of the device. They moved from very general knowledge of the device (as some kind of toy) to very specific knowledge that let them predict its behavior in detail. Subjects' theories included operational knowledge (e.g., pressing **FORWARD** makes the BigTrak move forward), syntactic knowledge (e.g., one must press a direction before a number), and "model" knowledge (e.g., the BigTrak has a memory that stores the steps previously entered). Operational knowledge could apply to the contents of the model as well as to the BigTrak as a whole. For instance, the knowledge that the **CLR** button erases the program is an operation on the program memory. The program memory itself is an element of model knowledge.

Most relevant to this discussion, we observed that between certain interactions with the BigTrak, subjects changed their theory of the device. A number of empirical generalizations seem to hold about the nature of these changes; see Shrager (1985) and Shrager and Klahr (1986) for more detail:

- The changes often took place when the BigTrak did something that the subject did not expect, or when the subject moved on to consider a new aspect of the device (e.g., the function of a new button).
- Instead of trying to determine in detail what led to a failed prediction, subjects usually observed what (positive behavior) took place and changed their theory according to that observation.
- The changes could be *non-monotonic*, in that they appeared to replace or delete knowledge that the subject believed before the reformulation.
- Both monotonic and non-monotonic changes in the learner's knowledge of the device were *non-uniform*. In some cases a single piece of knowledge was changed (e.g., another operation was learned for the device); in others, very large changes occurred (e.g., all of the numerical arguments took on new meanings).
- Changes often seemed to be informed by knowledge that lay *outside the domain* of the BigTrak. This included both particular knowledge, such as the numbering on the face of a clock, and abstractions like "programmable-device," which describe various devices that maintain state information in a memory. Changes that made contact with ex-

ternal domains usually brought *new concepts* into the BigTrak domain.

- Furthermore, the changes seemed to have a *coherent structure.* That is, instead of changing a number of small parts of the current theory in arbitrary ways, learners either made one change that affected many different aspects of the device, or else made several changes that were related in some principled way. For instance, learning that BigTrak is programmable led to a number of changes in the learner's operational and model knowledge. These are all related by the concept of programmability, although individually they are very different.

We concluded from our study that learners *incrementally combined abstractions* in order to formulate and reformulate theories of the BigTrak. Sometimes the abstractions were compatible with current knowledge and sometimes they were not. In the former case, such combination was a simple matter of adding the new knowledge and binding variables where necessary. However, introducing a new abstraction that was incompatible with the learner's current theory could lead to significant changes in that theory, including replacement of former beliefs. These findings led us to propose *view application* – a mechanism for learning through incremental combination of abstractions – as the principal mechanism of theory change.

3. View application in instructionless learning

View application is one component of our learning theory. To set this mechanism in context, we briefly describe the overall theory below. After this, we consider the representation of knowledge used in the model, the ways in which this knowledge is used, and the details of the view application mechanism. Finally, we consider an example of this process at work.

3.1 An overview of the IE model

In order to refine and validate our theory of instructionless learning we implemented IE (the Instructionless Experimenter), a computational model of this process. The system contains a number of different mechanisms for learning and diagnosing failures, bound together by a strategic control component. Many of the details are not relevant here, since our main concern is with view application. However, some idea of the model's structure will clarify the role played by that mechanism. We refer the reader to Shrager (1985) for other details of the model.

The input to IE includes a description of the BigTrak's visible structure, including wheels and a keypad with various keys; an auxiliary database of terms organized in an abstraction hierarchy, each represented by frames with various slots (e.g., the concept of `FORWARD` is a sort of `MOTION` with

DIRECTION = 0 (degrees)); and a set of independent abstract schemas called *views* for use in theory change. The program interacts with a BigTrak simulator by "pressing" buttons and observing the resulting behavior of the simulated BigTrak. This behavior is described in terms found in the auxiliary database, such as **MOVE FORWARD 9**.

Based on our psychological study, we have given IE the goal of associating at least one operation (as described later) with each physical feature of the device. The principal features of the BigTrak are its keypad buttons, but IE attempts to assign roles to the other features of the device – most of which are cosmetic. Subjects do this as well, sometimes finishing the task confused by cosmetic aspects. A complete theory of instructionless learning must explain how one determines the relevant features of a device, but this is not our focus here.

The system attempts to establish connections between operations and physical features through three data-gathering activities – *experimentation*, *exploration*, and *exercise construction*. IE formulates experiments to test a particular hypothesis in the theory – usually about one operation or model belief (e.g., that the BigTrak turns are numbered in minutes of arc). Experiments are accompanied by a specific prediction about the BigTrak's behavior if the hypothesis is correct. Explorations are used to elicit action from the BigTrak that can be analyzed and lead to theory changes. Explorations do not usually make a specific prediction, but they may predict some general behavior (e.g., pressing a button should make the BigTrak move in some manner). Finally, the program uses exercises to validate the entire theory. These incorporate a number of the known operations and, like experiments, carry a specific prediction about the device's behavior. IE halts when it has assigned a role to each physical feature of the BigTrak, and when it can construct and correctly predict the outcome of exercises that include all the operations in the learner's current theory of the device.

When one of IE's predictions fails or when it cannot run an exploration, the system tries to use causal analysis to determine why the failure occurred. It first attempts to assign direct blame for the failure, trying to assimilate the device's misbehavior into its current knowledge. This is done by using knowledge about the operations and the states of various model objects, backchaining from the observed behavior to a set of possible accounts for that behavior in terms of the current theory. Specifically, IE retains a trace of all the operations that were carried out in the course of the present experiment. It reads this trace from the end, basing its explanations on a set of rules that are specific to the operation involved.

For instance, if the BigTrak does not move at all in response to some sequence of button presses, IE first examines its memory of the operation sequence for the last operation used. Suppose that the last action was

to press the **GO** button. The function of this action was to run off the contents of the program that had been previously entered. The causal analyzer tries to determine what operations loaded the program and to ensure that the device was not in a state that might interfere with them (such as the BigTrak being turned off).

However, causal analysis is not our central concern. In the present implementation, IE's causal analyzer can formulate only limited types of explanation. The rules that control the generation of alternatives and backchaining were written to handle specific situations from our protocols or specific cases that were encountered in runs of the program. In complicated cases, we sometimes manually supply a causal analysis to the system.

If this process of blame assignment fails (which it often does), then IE tries to change the theory in order to *explain the failed prediction.* To this end, the system invokes the view application process, which we describe in more detail below. Basically, this involves incorporating an abstraction, suggested by the device's structure or behavior, into the theory.

3.2 The representation of theories and views

Now that we have considered the overall structure of the IE model, we can turn to the details of view application, the system's primary method for theory revision. This process operates by incrementally combining abstractions, called "views," into the learner's current "theory" of the device.

Our theory assumes that the abstractions called "views" exist in memory. Views lie outside of the current domain – narrowly construed as the BigTrak. Examples of views include the notions of memory, vector addition, "last in, first out" stack manipulation, clock time, and line-numbering in BASIC programs. None of these have any direct connection to the BigTrak domain, but all can be used to draw inferences about the device. Such an abstraction need not be abstract in all its aspects. Thus, one particular view of memory could specify the particular operations used to read and write the memory, but not the number or contents of memory elements.

IE represents both its current theory and the views as *schemas*, which are composed of one or more *frames* that refer to one another. A frame is a collection of *slots*, each of which has a name and a value. The values are list structures of terms and relations, which are defined by the auxiliary database mentioned above. The particular frames that compose a schema depend upon what is being represented. In the domain of programmable devices, these schemas include planning *operations*, *scripts* of routine usage, and physical *features*. Schemas also include *model objects*, which represent inferred components of the device. For instance, switches and memories are model objects, along with other abstract objects such

Table 1. An initial theory of BigTrak.

Scripts:
  Do an action-operation.
Operations:
  Type: **Action-operation.**
    Method: **Do each user-action in some code-sequence.**
    Function: **Do a Real-World-Action.**
  Instance: **Forward-9 is an action-operation.**
    Method: **Do each user-action in Sequence-1.**
    Function: **Do (MOVE FORWARD 9).** ;; a BigTrak action
Objects:
  Type: **Code-sequence.**
    Structure: **A list of user-actions.**
  Instance: **Sequence-1 is a code-sequence.**
    Structure: **Press CLR, Press FORWARD-ARROW, Press 9, Press GO.**

as "the class of buttons that cause immediate behaviors to occur." Model knowledge includes every object referred to in the scripts and operations, as well as objects referred to by other model frames.

We have simplified our presentation of views and theories, paraphrasing them in English for the sake of clarity. Table 1 presents a simple theory in terms of its (paraphrased) schema. This theory has one script and one operation. The learner knows that: `Pressing the button sequence:` **CLR FORWARD-ARROW 9 GO** `will cause BigTrak to` **MOVE FORWARD 9** `units`. Each operation has a `method` (e.g., `Press` **CLR FORWARD-ARROW 9 GO**) and a set of `functions` (e.g., **MOVE FORWARD 9**), which express the goals accomplished by the operation. There may be many independent functions associated with each method. All other frames have only a `structure` slot.

Table 2 paraphrases the `CODE.AND.STARTUP` view. (We will always indicated a view using capital block letters.) Essentially, this view says that there are two sorts of operations. One sort (`memory-operation`) sets the `program-plan` and then another (the `startup-operation`) runs the plan. Note that this view and its contents are not specific to the BigTrak, but rather represent general knowledge about programmable devices. The other views in the IE model have similar generality.

Table 3 shows the contents of the `CLEAR` view. Note that it does not specify programming or startup steps. These are not essential to the semantics of clearing the memory. What is crucial are the memory (`program-plan`) and the `clear-operation`, again a sort of `immediate-operation`.

Views do not refer to one another. However, they are "dovetailed" in that they are all represented using a common set of primitive terms. Although this is the usual AI solution to the problem of common reference, we view

Table 2. The CODE.AND.STARTUP view.

Operations:
  Type: **Memory-operation.**
    Method: **Do each user-action in some code-sequence.**
    Function: **Set the program-plan to a Real-World-Action.**
  Type: **Immediate-operation.**
    Method: **Do a key-press.**
  Instance: **Startup-operation is an immediate-operation.**
    Method: **Do a GO-keypress.**
    Function: **Do each act in the program-plan.**
Objects:
  Type: **Code-sequence.**
    Structure: **A list of key-presses.**
  Type: **Key-press.**
  Instance: **GO-Keypress is a key-press.**
    Structure: **Press a button whose label indicates CODE.AND.STARTUP.**
  Instance: **Program-plan is an action-plan.**
    Structure: **A list of acts, each of which is a Real-World-Action.**

it as a temporary simplifying assumption - albeit an important one. The particular contents of the view data base were collected from our protocol data. When we inferred that subjects used some view, we added that view to the database. Our final version of IE was provided with eight views:

CODE.AND.STARTUP: The notion of writing a program and running that program.

CLEAR: The notion of erasing the contents of a memory.

SIMPLE.CODED.DEVICE: The notion of giving a command and having the device carry out some action in response.

SIMPLE.PROGRAMMING: The notion of adding to an explicit memory.

COMMAND.ARGUMENT: The notion of commands that have syntactically associated arguments.

MODE.SHIFT: The notion of explicit modes and a way to shift between them. (Applying this view adds a mode check to some operation frames.)

BASIC: The notion of an addressable memory, including operations for placing things in the memory and for reading them out.

VECTOR.COMPRESSION: The notion of adding two directions and distances (as in vector addition), then replacing them with the result.

3.3 Using theories to generate interactions

The IE model uses the frames in its current theory to plan and execute interactions with the device in much the same way that top-down planning systems operate. Scripts contain goals that are satisfied by the functions associated with operations; alternatively, they may contain specific operations or operation types. Some operations specify physical interactions with the BigTrak (in most cases these are button presses). Other operations specify subgoals, sequences of subordinate operations, conditionals, or actions on model objects. For instance, one operation might load a value into a memory cell. The memory and its cells are model objects. Complex chains of planning and action can be represented by sequences of operations, rooted in scripts, and leading through subordinate operations to button presses. As the interaction planner expands operations, it formulates a prediction by collecting the functions of the operations that are used. The resulting prediction includes both external behaviors of the device, such as movement, and internal actions, such as the storage and retrieval of information from memories.

Consider the simple initial theory displayed in Table 1. Beginning at the script we see that it states one should "`Do an action-operation.`" The only action-operation is `Forward-9`. Interaction planning pursues the `Method` slot and records the `Function` slot as an expectation. In this case the `Method` says to "`Do each user-action in Sequence-1`" and the `Function` (i.e., what this method accomplishes in the BigTrak domain) says to expect the BigTrak to "`Do (MOVE FORWARD 9).`" The user-actions for our method are obtained from the `Structure` slot of `Sequence-1`. These are the keypresses: **CLR**, **FORWARD-ARROW**, **9**, and **GO**, which are done at this time. Given a more complex theory, one can see how `Function` slots could lead to further expansion and thus to a richer set of expectations by examining the `Memory-Operation` and `Startup-Operation` frames in Table 2 (pretending for the moment that this is a theory rather than a view). In this case, the functions of these operations get and set the structure of the `program-plan` object. More complex theories contain more complex objects and operations, possibly including state variables and conditions, thus leading to more complex experiments and predictions.

3.4 The process of theory modification by view application

Now that we have considered the representation of views and their use in experimentation, we can examine their role in altering theories. The principal assertion underlying view application is that theories are constructed by incrementally combining abstractions encoded as views. Below we briefly discuss the selection of views and then detail the process of

Table 3. The CLEAR view.

Operations:
  Type: **Immediate-operation.**
    Method: **Do a key-press.**
  Instance: **Clear-operation is an immediate-operation.**
    Method: **Do a clear-keypress.**
    Function: **Set the program-plan to NIL.**
Objects:
  Type: **Key-press.**
  Instance: **Clear-keypress is a key-press.**
    Structure: **Press a button whose label indicates the view CLEAR.**
  Instance: **Program-plan is an action-plan.**
    Structure: **A list of acts, each of which is a Real-World-Action.**

view application. We also include some comments on the assumptions that underlie this process.

3.4.1 View selection

There are a number of ways that a particular view can be selected. These include recognizing the view associated with the label on a key (e.g., "CLR" suggests CLEAR); noticing that some action of the BigTrak is similar to the action that one would obtain from some particular view (e.g., movements are mentioned in a "vehicular" view); recognizing that the BigTrak is "doing the same thing as before" and selecting PROGRAMMABLE.DEVICE in response. Another important selection mechanism uses the contents of the learner's current theory to trigger view applications. For instance, if the theory contains a memory, then the learner might try to apply the CLEAR view, guessing that there is a way to clear the memory.

Other selection mechanisms certainly exist. Our human subjects seemed to retrieve useful views in a variety of ways, and we implemented such selection mechanisms as needed to account for their behavior. However, we do not believe the set of techniques in IE is comprehensive, nor do we have a principled theory of view selection. This is an obvious direction for future research, but our main concern here is with view application.

3.4.2 View application

Once IE has retrieved a plausible view from memory, it carries out the following five steps in order:

1. Create a copy of the selected view, duplicating all frames in the view along with their contents. We will call this copy the *base.*

2. Merge the current theory with the base. This involves matching all frames in the theory schema for which a correlate can be found in the base and then copying the remaining unmatched frames from the current theory into the base.

This matching process relates frames from the current theory to frames in the view. Each frame has an abstract type (e.g., `memory-operation`, `program-plan`), and each major frame type (scripts, operations, etc.) is independently matched, so types cannot mix. However, multiple frames in a view can have the same type resulting in multiple possible matches. If matching is ambiguous, IE selects one of the possibilities at random; we will discuss this design decision later. There might also be unmatched frames of a particular type. These frames are simply added to the base.

3. Coerce all of the frames so that they are appropriate children of their new parents.[1]

As a result of step 2, some instance frames that were children of a particular frame in the old schema may now be children of a different frame from the new view. Numerous heuristics are utilized in this coercion step; these are cued by the syntax of the frame's slot contents. Suppose that an operation contains some relation between movement and pressing a particular button; say the operation `FORWARD` means approximately: "Pressing **FORWARD** makes the BigTrak immediately move forward." Further suppose that the view imports the concept of a program memory, in which a list of movements can become a program plan, and of programming (a `memory-operation`). The meaning of this latter concept is approximately: "Put some action instruction into a memory cell." For instance, if `FORWARD` is a kind of `immediate-operation` and the new parent frame (imported from a view) is `memory-operation` (replacing `immediate-operation`), this step will coerce `FORWARD` so that it is an appropriate child of `memory-operation`. For instance, the coercion might result in an operation that saves the forward movement in a memory to be executed later, or that moves forward some remembered number of feet. Another possible interpretation is that executing `FORWARD` repeatedly will result in the repetition of the indicated movement. Some of the heuristics used in the coercion are specific to the primitives that compose all representations, both general and specific (e.g., `list of`). Other coercion heuristics refer to domain-specific elements (e.g., `forward movement`) and will differ for different domains.

[1] We use the term *coercion* to suggest data-type coercion in which, for instance, real numbers are coerced to integers (by rounding). In essence, this step is "coercing" frames whose type (i.e., whose parent frame) has changed so that they are valid instances of the new parent/type.

4. Force frames that are not fully instantiated to be filled in with various features and objects from the theory.

Views contain some frames that are not fully instantiated. Some of these will have been instantiated in the match step (2), but some may still be unspecified. Continuing the example, the CLEAR view contains a program-plan and a clear-operation, neither of which is fully instantiated. Suppose that the merge operation has bound the program-plan appropriately, but that there is nothing for the clear-operation frame to match in the current theory schema. In this particular case, cues in the uninstantiated clear-operation frame constrain its binding to a feature (e.g., a button) that "indicates" the CLEAR view. This view relies on the lexical semantics of the button labels to constrain selection.

As we mentioned earlier, the frames in a view are described at various degrees of abstraction. For instance, in the CLEAR view, there is an object called program-plan whose structure is given only as some unspecified list of real-world-actions; there is also an operation called clear-operation whose action is to nullify the structure of program-plan. When IE applies this view to the current theory, it copies program-plan (or matches it) into the target and instantiates the new program-plan's structure with some object out of the theory schema.

5. Finally, IE ensures that each object frame plays a role in only one operation.

This implements a "single-function" assumption, which is appropriate for the BigTrak, but which may not hold in all complex systems. Suppose that the **CLR** button (as an object) is used in some operation Op-1. Further suppose that applying the CLEAR view forces **CLR** to be bound in the new clear-operation by step 4. In this case, the heuristic in step 5 would remove **CLR** from its place in Op-1. Certain changes in the operation structure must be reflected in the script frames and other operation frames that call upon the modified operations as subordinates. This step might entail breaking structures into multiple entries where a single operation type has now been replaced by a number of operation types. For instance, there are now two operations: clear-operation and Op-1 where previously there was only Op-1. Scripts or other operations that used Op-1 will be changed to call upon both the new clear-operation and the revised Op-1. This editing is done only where the modified operation is explicitly named.

3.4.3 The independence of theories from views

Some additional comments will help to clarify the assumptions of view application. First, only one view is applied in each reformulation step and the learner's theory is permanently changed as a result of this view

application. Second, the new theory is not connected to the abstraction (the view) used in the change. This last point distinguishes view application from many classification models of learning. Instead of classifying the BigTrak as an instance of a `programmable-device` and storing this connection, IE uses the abstraction of programmable device *once* to modify its theory of the BigTrak. After this occurs, the system does not record any trace of the reformulation or any connection between the view and the theory. This means that IE cannot draw on such knowledge later on, and our human subjects did not appear to do so either. For instance, once subjects used the concept of BASIC line-numbering, they usually moved on to other views rather than importing additional knowledge of BASIC.

3.4.4 Non-monotonic theory changes

Our psychological data indicated that subjects made non-monotonic changes in the process of learning about the BigTrak. View application can lead to such non-monotonic changes in several ways. First, new frames brought in with a view can replace conflicting previous frames. (This differs from conflict resolution at the frame matching level, which is arbitrarily decided.) Second, a frame can be associated with a new parent type; the coercion step (3) will then change the frame's contents. IE's experimental strategy can also make the system *appear* to forget certain knowledge. Recall that interaction planning and execution always begins at the script frames of the current theory. A given script contains a list of functions, operations, and operation types. As a result of the changes made by view application in scripts and other operations, some operations may no longer be mentioned in the extension of any script. That is, during execution planning, some operations are not accessed. These operations will never appear in any use of the BigTrak. The operations are essentially lost, but their frames still exist in the current theory. Before these operations can be reused, they must be reattached to the extension of some script. This can occur when view application leads to a new script for the device that includes these operations, or because they are included in the extension of other operations (in step 4). Since these frames are retained in the theory schema, they can still play a role in matching and instantiation, and thus may come into play again during later learning. Our psychological data are ambivalent on this aspect of instructionless learning. People sometimes appear to relearn operations and sometimes appear to recover old operations. This fact led us to retain inaccessible frames.[2]

[2]This explanation of non-monotonicity is similar to Simon's (1976) explanation of how EPAM-like theories can fail to recognize a new stimulus, even though it has been previously stored. "...if one meets, and becomes acquainted with a new John Smith, access to information about another John Smith one had known previously may be lost, *or at least become more difficult to recover.*" (Our italics.)

3.4.5 Unreasoned decision making in view application

View application is an inherently ambiguous process. Steps 2, 3, and 4 in the above algorithm all involve heuristic decisions about frame matching and about the bindings of various variables. However, in contrast to many problem-solving models (e.g., Laird, Rosenbloom, & Newell, 1986), *no problem solving for decision-making purposes takes place during view application.* If IE cannot distinguish between a number of options, it selects one at random. If conflicts arise in step 2, the model randomly resolves them as well. This may seem odd, since further reasoning might resolve the conflict in a principled way. However, our theory makes the strong claim that *in low cost situations learners seldom think very hard* about such decisions. Instead they rely upon interaction with the device to work out mistakes. If they make a poor decision, they will discover it during interaction with the device. Our protocol subjects seemed to make reformulation decisions without much thought, and they made bad decisions as often as they made good ones. Therefore, random resolution seems to be a reasonable model of our subjects' behavior.

A further strong claim of our theory is that the learner retains *no memory of the decisions made during view application*, so no backtracking through the space of theories is possible. In our experiments, humans never backtracked through the space of ambiguous views and bindings. They relied upon experimentation with the BigTrak to reveal errors, and then used information from the failures to patch the theory, perhaps leading to new view applications.

In sum, we argue against the notion that reasoning and problem solving play a central role in theory formation or reformulation per se. Instead we propose that theory formation and reformulation take place primarily by architectural mechanisms (e.g., view application) that do not involve problem solving. The learner relies upon the structure of the whole learning interaction to find mistakes and (eventually) to generate an appropriate theory. The success of this strategy depends crucially upon the low cost and high rate of interaction in the instructionless learning setting. In higher cost situations, such as learning about a live air-traffic control system, theory formation would presumably take place in precisely the same automatic way. However, the learner might take care to consider the implications of its theories before actually putting them into practice, and so reject theories that would lead to costly or dangerous experiments.

3.5 An example of view application

Now that we have examined the view application algorithm and the assumptions it relies upon, we turn to an example of the process. The

Table 4. The theory after applying CODE.AND.STARTUP. The righthand column indicates that the entry in this row came from the view (**V**), the theory (**T**), or a combination (**VT**). The "(5)" indicates that step 5 of view application changed this entry.

Scripts:	
Do a memory-operation followed by an immediate-operation.	**T(5)**
Operations:	
Type: **Memory-operation.**	**V**
Method: **Do each user-action in some code-sequence.**	**V**
Function: **Set the program-plan to a Real-World-Action.**	**V**
Instance: **New-Forward-Operation is a memory-operation.**	**VT**
Method: **Do each user-action in sequence-1.**	**T**
Function: **Set the program-plan to (MOVE FORWARD 9).**	**VT**
Type: **Immediate-operation.**	**V**
Method: **Do a key-press.**	**V**
Instance: **Startup-operation is an immediate-operation.**	**V**
Method: **Do a GO-keypress.**	**V**
Function: **Do each act in the program-plan.**	**V**
Objects:	
Type: **Code-sequence.**	**VT**
Structure: **A list of key-presses.**	**VT**
Instance: **Sequence-1 is a code-sequence.**	**VT**
Structure: **Press CLR, press FORWARD-ARROW, press 9.**	**T(5)**
Type: **Key-press.**	**VT**
Instance: **GO-keypress is a key-press.**	**V**
Structure: **Press GO.**	**V**
Instance: **Program-plan is an action-plan.**	**V**
Structure: **A list of acts, each of which is a Real-World-Action.**	**V**

example here demonstrates several view application steps taken from IE executions that are more fully presented in Shrager (1985). We begin after the system has obtained some knowledge about the BigTrak, represented by the theory in Table 1. IE knows a single BigTrak operation: `Pressing the button sequence: CLR FORWARD-ARROW 9 GO will cause BigTrak to MOVE FORWARD 9 units.` Note that at this point **CLR FORWARD-ARROW 9 GO** is a single undifferentiated sequence of button-presses. That is, the button presses are not the result of setting subgoals and executing other operations. However, this sequence of button presses is accessible to the learner.

IE examines the contents of all of the operations in the theory (in this case, just the `forward-9` operation). It considers applying views that are suggested by the labels of keys or by some pattern in the operator contents. The use of the **CLR** button suggests applying `CLEAR`, **GO** suggests applying the view `CODE.AND.STARTUP`, and a command arrow followed by a number suggests applying the view `COMMAND.ARGUMENT`.

In this run, IE selects the CODE.AND.STARTUP view (paraphrased in Table 2) and incorporates it into the existing theory of BigTrak, giving the revised theory shown in Table 4. This modification does not change the system's external interactions with the BigTrak. The main difference is that the button sequence, initially interpreted as the method of a single operation, is now parsed into two separate operations: a programming step and a startup step (pressing **GO**). But this step has introduced several *new concepts* into IE's theory of the BigTrak, including memory and memory-loading operations, the distinction between memory operations and immediate operations, and the notion of an action sequence. These concepts will affect future interaction and learning in a number of ways: they provide suggestions for later view applications; they provide frames into which aspects of later view applications can be merged; and they will be used in the operation of IE's causal reasoning component to interpret BigTrak behaviors.

The system next applies the view CLEAR (paraphrased above in Table 3), as suggested by the occurrence of the **CLR** button in the `new-forward-operation`. This time, instead of adding a new type of operation, the CLEAR view's `immediate-operation` type is bound to `immediate-operation`, a component of the theory learned in the preceding application. Similarly, the `program-plan` matches the existing `program-plan`, and so forth. A new operation instance is generated for `clear-operation` and the `clear-keypress` is instantiated with the **CLR** button. This is removed (by step 5 of the view application process) from the previous `code-sequence`.

At this point, IE proceeded to apply the concept of multiple-step programming. After this, the system incorporated a notion similar to programming in BASIC. In this view, slots in the program memory are addressed by numbers. In the revised theory, action operations (e.g., FORWARD) have implicit distances of one foot, and the numerical argument refers to the position in the memory in which this operation is placed. For instance, pressing **FORWARD-ARROW 9** puts (**MOVE FORWARD 1** (foot)) into location 9 of the program memory. **GO** causes the memory to be run from the beginning until the end. Thus, applying this view has non-monotonically changed the semantics of operation arguments (the numbers) to represent line numbers in memory. The memory is also reformulated by this view application so that it can be addressed by the numerical arguments. The resulting theory is far too complex to present here, but Shrager (1985) provides a detailed description. Although this is *not* a correct theory of the BigTrak, it *is* coherent and reasonable.[3]

[3] At this point the knowledge in IE's causal reasoner is insufficient to handle prediction failures for a theory of such complexity. The program fails for this reason.

4. Evaluating view application

Now that we have described the view application algorithm and seen it in operation, we can evaluate the mechanism along a variety of dimensions. Below we analyze view application in terms of search through a space of theories, showing that it constrains this search in significant ways. After this, we consider the manner in which view application explains results from our studies of human instructionless learning. Finally, we consider the generality of the mechanism.

4.1 View application as constrained search

The view application process is rather complex, and it is only natural to question whether the benefits of this mechanism outweigh the price. To explore this issue, we will carry out a search-based analysis of view application.[4] In search terms, each view application step can be viewed as a single operator that generates a new successor state in the space of theories. Each such step may add a number of frames to the existing theory. Given an initial theory and a set of views, one can use view application to exhaustively generate all theories down to a given level. We will call this the *view application theory space* (or just the *view space*). We will compare this space to an alternative one in which each operator adds only a single frame to the current theory. We will call this the *simple theory space* (or just the *simple space*).

View application assures that *all* theories in the view space are "coherent." For example, an incoherent theory of the BigTrak might state that the device is programmable but not mention that it has a memory. Since views organize theory changes that are related to one another, view application (together with properly written views) ensures that such inconsistencies never occur. A simpler learning scheme that ignores the constraints inherent in views could easily generate an incoherent theory. Thus, the most important difference between the view space and the simple space is that the former contains only coherent theories, while the latter contains many incoherent ones as well.

Coherence has both practical and scientific value. The assumption that theories are coherent considerably simplifies the rest of the IE system. For example, one type of incoherent theory contains unbound frame variables; another type mentions frames that do not exist. Since view application

[4] Although IE does *not* search the theory space in the normal sense of breadth-first or depth-first traversal, a search analysis can still lead to useful insights. View application does lead the system through a space of theories, but the method is much closer to hill climbing than to more thorough search strategies. This observation is due to Pat Langley (personal communication).

forbids these possibilities, IE does not need to check for them. One could certainly include the necessary tests, but this would greatly complicate the system. More important, adding such tests would distribute the process of theory generation throughout IE, rather than having the process done entirely by view application. Our psychological data suggest that new theories are constructed all at once, rather than in the piecemeal fashion required in the distributed version. Subjects rarely change their theory in the middle of an interaction with the BigTrak.

The differences between the two theory spaces is further revealed by examining their respective branching factors. Briefly, the view space is shallower and wider than the simple space, but it will be useful to consider the extent of this effect. Let us assume there exist six views in memory, each containing two object frames and two operation frames. One such view is the concept of a stack, whose objects are HEAD and REST and whose operations are PUSH and POP. We may ignore script frames, since these average less than one script per view. We may also ignore physical feature frames, because they are fixed for any given device.

One can think of a learning sequence as a series of E view application events. At each of these events any one of the six views can be applied, giving 6^E possible ways of combining views. However, in general there are many ways to incorporate a given view into the current theory. Because view application never deletes frames, the theory can only become more complex at each step. Even when the same view is applied more than once, the complexity of the theory can increase, since specific matching and instantiation decisions can be made in different ways. As a result, the branching factor itself increases with the depth of search.

The very first view application is simple. Since there will be no object or operation frames to match in the learner's current theory (algorithm step 2), all frames from the view will be added to form the new theory and no coercion will have to be done in step 3. If there are variables in the view that must be instantiated with parts of the current theory (step 4), then there is probably ambiguity. For instance, the learner might have to decide which of several buttons actually clears the device's memory. However, this only multiplies the complexity of view application by a constant factor. Therefore, the number of ways to transform the initial theory (with no operations or objects) approximates the number of available views – six in this case.

Each of the resulting theories contains two operators and two objects. During the second view application event, one will either match a view frame to a theory frame or add the view frames (unmatched) to the theory (algorithm step 2). As we have noted, the number of possible matches increases with the number of frames in the theory. Let us continue with

the assumption that there are two object frames and two operation frames in each view, and suppose N is the number of operation and object frames in the current theory. In this case, there will be one way to incorporate each view into the current theory in which no frames match, there will be $2N$ ways in which two frames match against each other, and there will be $N(N-1)$ ways in which only one frame matches. We then square this number to take both objects and operations into account, giving the expression

$$M = (1 + 2N + N(N-1))^2 = N^4 + 2N^3 + 3N^2 + 2^N + 1$$

for the number of ways one can match a view against the theory. In other words, the number of matches M increases on the order of N^4.

If we further assume that the coercion and instantiation decisions (algorithm steps 3 and 4) can be done in two ways for each frame, there will be approximately 2^M ways of coercing and instantiating the matched frames. However, in practice the views provide a great deal of constraint; for instance, we saw in Table 2 that **GO-Keypress** was constrained to bind with some key that indicated the `CODE.AND.STARTUP` view. Therefore, we will conservatively estimate that this ambiguity increases the branching by a factor of 10.

Combining these factors, one can compute the total number of possible theories at the second level of the search tree. Assuming N is 2, we have $M = 2^4 = 16$ match/add decisions. Multiplying this number by ten gives 160 theories, and multiplying this by six views comes to 960 successors for each theory at the first level. Since there were six of these, we have a total of 5760 possible theories after two view application events. The general formula for the kth level is difficult to specify analytically, since alternative theories at a given level contain different numbers of frames. However, a very conservative estimate is $60k^4$, where k is the number of view applications that have occurred. Even this estimate makes the conservative assumption that there exist only six initial views with two object frames and two operation frames each.

Naturally, the branching factor in the simple theory space is much smaller. In this space, the number of alternatives at each level is determined by the number of frames in all the views ($6 \times 4 = 24$ given our assumptions), multiplied by the number of ways that a given frame can be combined into the current theory by addition or merging. This latter number will depend upon the number of frames in the theory, which remains the same at each level (if the frame is combined with an existing frame) or increases by one (if the frame is added). Given the set of views we have been assuming, the second level of the simple space contains only 96 theories, much less than the 5760 number we obtained for the view space.

However, recall that one step in the simple space accomplishes much less than a step in the view space. On the average, one view application step adds the same number of frames as four steps in the simpler space. This means the appropriate comparison is between the second view level and the eighth simple level. The latter space contains approximately 124 million alternative theories after eight simple operations, many more than the 5760 in the view application space. This difference only increases with deeper levels and more complex views.

This clarifies the nature of the two theory spaces. At equivalent levels, the simple space contains all the theories contained in the view space, as well as many others. The vast majority of these theories are incoherent in that they violate constraints that would be obeyed by the view application process. Despite the size of the view space, the simple "hill-climbing" strategy used by IE is sufficient precisely because all of the theories contained in this space make some sort of sense. View application lets the system make large, organized leaps in its knowledge about a device, greatly constraining the process of theory construction.

4.2 View application and human instructionless learning

In section 2 we described several generalizations from our psychological experiments in instructionless learning. For instance, we noted that subjects learn by observing the positive behavior of the device[5] when their predictions fail, rather than backtracking or doing careful causal reasoning about the failure. This led us to propose a simple reformulation mechanism embedded in a learning system that takes advantage of the interaction with the device to uncover erroneous theories. Subjects made complex non-monotonic changes to their theories using knowledge that seemed to lie outside the domain of the BigTrak (e.g., knowledge of vector addition). This led us to propose view application as the mechanism for theory reformulation by incremental combination of abstractions.

The above results – that the view application theory space is much shallower and more dense with coherent theories than the simple theory space – permit us to make further comparisons with the psychological data. Human learners apparently do not try very many different theories. They execute only about 52 interactions with the BigTrak. We do not know if human learners devise incoherent theories of the device and then reject them without an interaction. However, the mean time between interactions

[5]In saying that people use the device's "positive" behavior, we mean that they usually observe and analyze what the device actually does, rather than analyzing how the device's behavior differs from what was expected.

is just 28 seconds, so it seems unlikely that people are internally considering very many alternative theories.

When IE is playing the role of a cognitive model, we try to match a particular human protocol and so we make specific decisions about which view to apply at each step and about which frames to match between the view and the theory. In this mode, all of IE's learning and causal reasoning mechanisms are active in addition to view application. This permits the system to simulate the complex interaction of mechanisms that play a part in human instructionless learning. Elsewhere (Shrager, 1985) we argue that as a cognitive model IE's knowledge (captured primarily in the views) can account for the expected theories of our subjects. Moreover, the model will not overpredict by generating theories that we would find unnatural for human learners. IE does not actually simulate long learning sessions by itself because its causal reasoner is very limited. However, without invoking causal reasoning, we can make IE track subjects' behavior in sessions of learning about the BigTrak through as many as six view application steps, involving up to 12 interactions with the BigTrak. If we manually give IE the results of causal reasoning, it can track sessions of up to three times this length (about 30 interactions). Since human protocols contain about 52 interactions with the BigTrak, some of which do not correlate with theory changes, IE comes close to modelling entire learning sessions.

Some other generalizations arise from view application learning that correlate closely with our human protocol data. View application generally leads the system to incorporate mechanistic notions into its theory because such mechanisms are part of the views. IE is *not* given the goal of discovering a mechanistic theory of the device under study, but it does this anyway as a side effect of view applications. As a result, IE induces a mechanistic model for the device. This mechanistic model plays a role in causal reasoning at failures and in later view application. The induced mechanistic model thus has a direct impact on later learning.

The increasing complexity of theories leads IE to engage in increasingly complex interactions with the device. We saw that the length of the "exercise" interactions generally increased in length over the session. IE also hypothesizes increasingly complex internal machinations in the device model. For instance, some devices are not programmable in the sense of putting in many steps of the same type before executing any of them. A digital watch is such a device. However, if IE applies the `BASIC` view while studying such a device, it will expect to be able to store steps in a program memory, which is wrong. Thus, theories can become more complex without necessarily becoming more correct. IE will retain an incorrect theory until some view application suggests a different function for the aspects of the device that were incorrectly associated with programmability.

4.3 Generality of view application

The main constraints on IE's generality are its representation for the BigTrak theory and its initial knowledge base of views. The choice of scripts, operations, and model objects is specific to interactive complex devices, but is still quite general. This framework was drawn from an approach that Moran (1981) applied to a mail system, but other machines can easily be represented. In addition to using this representation for the BigTrak we have used it with simplified models of a compact-disc player, a microwave oven, and a digital watch.[6] Although this is a fairly small set of devices, there seems to be no reason, in principle, that IE could not go further with an appropriate set of views.

The IE model also makes a number of assumptions about the nature of the device. For instance, it assumes that the device exhibits deterministic behavior, an assumption that our subjects seemed to share. Deterministic behavior is implicit in the particular views that IE uses, and this leads to deterministic theories. But there is no intrinsic constraint that requires determinism, and one can create a non-deterministic theory by simply applying a non-deterministic view. Such a view might contain internal operations that select among subgoals randomly. For instance, one can create a view in which commands are placed into random locations in the program memory. This could be merged with the learner's current theory by view application, leading to a theory that has random command storage. One would have to add some specific performance rules to interpret non-deterministic characteristics, but this does not affect view application.

Another assumption made by the IE model is that each operation has a single associated function. It is more difficult to write views that violate this assumption, since it is crucial to view application. The single-function assumption assures that appropriate changes are made in the scripts of the theory when changes are made in operations (step 5 of the algorithm). However, even devices with multiple-function operations can evoke only one function at a time. One can easily simulate this situation by dividing the complex single operation into multiple operations and adding state information to the model objects. The separate operations would be triggered according to the mode that the device happens to occupy at the time. Therefore, with some effort, one can represent a device that has multiple functions associated with its operations.

[6] Simulating these devices entailed making certain displays and outputs into explicit actions, and this removed much of the richness of the devices' behaviors. For instance, our simulations represent the musical output of a CD player, the ejection of the disc, the movement of the BigTrak, and the BigTrak's generation of beeping sounds as behaviors of precisely the same kind, using symbolic terms like `BEEP` and `SONG-1`.

The most important assumptions made by the IE learning model are that the device under study is interactive, that interactions will produce rich behaviors to cue view applications, and that the cost of interaction is low. IE will not do very well in learning how to operate an air-traffic control system; perhaps it is more correct to say that the passengers will not do very well. The simplicity of view application depends directly on the ability of the learning system to work out erroneous decisions by interaction with the device, and to learn by observing what occurs when predictions fail.

5. Discussion

In this section we compare view application with other knowledge-rich learning models and models that work in similar domains. We close by considering some directions for future work and by summarizing our results.

5.1 Related approaches to knowledge-rich learning

IE generates a theory of a complex device through knowledge-based theory formation, and view application plays the principal role in this process. This mechanism is related to a variety of other techniques described in the AI literature, including analogy and categorization. Below we compare and contrast it with these earlier frameworks.

In the analogical approach (Gentner & Gentner, 1983; Greiner, 1985), one selects a *source* schema from the schema library and carries over knowledge into a *target* schema. At first glance this seems very similar to view application, but there are a number of differences. First, analogy maps between two instances, whereas view application maps an abstraction (a view) onto an instance (the device theory). Second, most analogical learners are additive – they only add structure from the source into the target; in contrast, view application can both add knowledge and reformulate existing knowledge by coercion. Finally, analogical methods typically associate the source concept with the target concept by finding some common abstraction. In view application, the source begins as an abstraction. However, a view is not an abstraction *from* the current theory; rather, it contains more or different information that can be applied to the theory.

Within research on analogy, Bott's (1979) work on learning the Unix editor and Burstein's (1983) work on incremental debugging in learning BASIC are most similar to the current effort. Like IE, both Bott's and Burstein's systems incrementally incorporate knowledge through successive analogies. However, both depend upon a teacher's advice, whereas IE reformulates its theories by observing device structure and behavior. Also, Burstein's program debugs the current theory when problems arise; in

contrast, our model repairs problems mainly by asserting a new view. As a result, IE makes broad jumps in its theory space, directed by the features and behaviors of the device it is observing. This also departs significantly from approaches that use "causal theory reformulation" (e.g., Hammond, 1986), which work primarily by accounting for failed predictions.

Another related approach involves the notion of categorization. In this framework, one infers knowledge about the target by classifying it as an instance of some more general source concept. The script applier mechanism of SAM (Schank & Abelson, 1977; Cullingford, 1981) is one example of a categorization scheme, and view application is similar in many respects. However, scripts organize information about story events, whereas views organize knowledge about operations, objects, and scripts that make up complex systems. Scripts typically represent sequences of events, leading to a predictive or interpretive processing mode; special reasoning is invoked only when expectations are violated. Therefore, one continues with the current script until some problem occurs, without dealing with combinations of scripts or plans. In contrast, IE attempts to understand a device's structure in its entirety; this means it must apply an entire view at once and thus deal with the problem of incrementally combining views.

Wilensky (1983) has provided one account of conceptual combination using script-like representations. His PAM/PANDORA model maintains multiple goals, each of which might trigger different plans. A meta-planner embodies heuristics for resolving goal conflicts and other problems. Thus, problem solving (meta-planning) takes place in order to combine goals and plans. The theory of view application explicitly excludes this technique, making decisions arbitrarily and then relying on experimental interactions to uncover mistakes. Wilensky's approach is also similar to IE in that it continually revises its interpretation of a story, refining its analyses according to later findings. However, in PAM these refinements are required because of missing information; this differs from IE, which must reformulate its theories because its unreasoned decision-making processes can make the wrong choice.[7]

Most researchers in the script inference tradition have paid a great deal of attention to "getting the inferences right." In contrast, our theory claims that in many circumstances, the learner need not expend the time and energy to ensure correct inferences, since problems will be worked out during interactions. The ability of our learner to succeed without deeply reasoned theory changes and without backtracking results from the low cost of interacting with the BigTrak in the instructionless learning paradigm.

[7]Furthermore, Wilensky modelled "idealized" reasoners, whereas IE simulated the behavior of actual subjects.

The EG system (Dietterich & Buchanan, 1983; Dietterich, 1984) also bears a close relation to the current work. This project involves experimenting with (and forming theories about) a live Unix system in the same way that IE experiments with (and forms theories about) the BigTrak. Since Unix has a large number of internal state variables, inferring these variables is one of EG's major goals. State variables play the same role for EG that a device model plays for the IE system. Without a device model, state dependencies in the BigTrak would make its description quite complicated.

However, EG also differs from IE in some important ways. The first divergence involves methods for inferring state variables. As we showed in section 4.1, the introduction of even a simple device model leads to the possibility of making incoherent theory changes. View application resolves this dilemma for the IE system, but EG actually searches a complex theory space with a syntax similar to that of programming languages. It makes no attempt to incorporate theories by analogy or script-like instantiation. Second, IE relies upon the experimental interactions to uncover problems; this lets it avoid much causal analysis or theory-driven interpretation of data. In contrast, Dietterich's approach relies heavily on such a data interpretation module. Finally, experimentation plays a specific role in EG with respect to making search decisions. Since IE is not explicitly searching a theory space, it uses experimentation to validate theory changes rather than to direct them. In this respect, IE is closer to Burstein's use of experimentation than to Dietterich's approach.

5.2 Directions for future research

Although we believe view application constitutes a significant contribution, clearly more work remains to be done on instructionless learning and methods for incremental conceptual combination. Below we discuss a number of promising research directions.

The weakest aspect of IE is the interaction between the causal reasoner and the view application mechanism. In section 4.1, we saw that the system may propose an arbitrary number of model objects in a device as a result of repeated view applications. The quest for parsimony seems to prevent humans from taking such actions, and this is an obvious criterion to include in future theory formation systems. Certain view application steps should only be made when they are necessary to account for observed effects. Dietterich's EG system incorporates such a heuristic constraint.

Despite our contrast between knowledge-based and search-based learning methods, these two approaches are not mutually exclusive. One can imagine a combined approach that used view application to take large steps

in the theory space and a search method to take smaller steps. The view application process would impose an overall structure on the search space, and a search mechanism could then explore variations on this structure.

Moreover, view application suggests an approach to revising the structure of this space over time. Although IE currently uses views only to form concrete theories, it could easily be altered to combine simple views into more complex ones. Rather than forcing its theories to be completely instantiated, the system would simply leave open slots in frames that are not fully instantiated (removing step 4 of the algorithm). This process would lead to new abstract views, which would provide more powerful (and more constrained) "operators" for future search. Thus, view application does more than account for knowledge-rich learning; with modification it should also explain the acquisition of the knowledge base itself.

Our present research has gone in a different direction. IE assumes that all views and theories are composed from a common set of primitives. This assumption pervades artificial intelligence, and is most explicit in systems based upon conceptual dependency theory. However, this is an unreasonable assumption for systems with a large amount of complex knowledge. It is difficult to find a natural way to express some concepts in an enforced ontology of terms and relations. For this reason, we feel that learning systems should be able to combine schemas (views, device theories, etc.) that are expressed with different terms and relations. Therefore, we have been studying the problem of "framework alignment." That is, how can views that are cast in different terms and relations be sensibly combined? The answer to this question has implications far beyond the task of instructionless learning, and we hope to report on our progress in future papers.

5.3 Summary

In our psychological studies of instructionless learning about complex devices, we found that people sometimes make large coherent changes in their theory. We interpreted these reformulations as cases in which the learner chose to *view the device in a different way* – specifically, as an instance of some existing abstract knowledge structure. This insight led us to implement a theory formation system (IE) whose principal mechanism is *view application*, which carries out this abstract reinterpretation (or reviewing) process.

View application implements incremental conceptual combination between abstract schemas, called views, and a concrete theory of the device under study. The mechanism works by classifying the device as an instance of an abstract category, based on the device's structure, its behavior, or as suggested by the learner's current theory. The learner then selects an

abstract schema that is representative of the category and reformulates the current theory in terms of that schema. No problem solving takes place during view application itself. Ambiguities are arbitrarily resolved and then worked out during later interactions with the device, either by further view applications or by other aspects of the complete learning system. The present theory does not specify how views are learned, merely how they are *used* in learning.

We designed IE and view application for the task of understanding complex devices, and it has been most extensively tested for the BigTrak, a programmable toy tank. This was the device that we used in our psychological studies. However, we have since tested IE on other devices, such as a simulated compact-disc player, a digital watch, and a microwave oven. Some additional views had to be added to the system to let it handle these new domains, but the basic mechanism of view application remained unchanged.

The theory that IE constructs by view application becomes more complex as learning progresses, although it will vary in correctness. The resulting theory generally contains a mechanistic model of the device, but this results from the particular views we provided, rather than from the process itself. In principle, IE can learn about any device for which it has appropriate views and for which the cost of interaction is low.

One clear benefit of view application is that it lets the learning system make large, coherent jumps in the theory space. Learning in this way compares favorably against search in a theory space that is connected by finer grained operators. The view application theory of learning also matches our psychological data better than the more atomistic search model. The theory suggests that simple knowledge-rich learning can succeed in very complex domains, provided the learner is given sufficient background knowledge, and provided he can interact with the system under study. IE is such a learner and view application is its principal mechanism.

Acknowledgements

This research was conducted at Carnegie-Mellon University and Xerox PARC; it was supported by an IBM fellowship and by grants from NSF (BSN81-12743) and the Spencer Foundation. David Klahr, John Anderson, and Herbert Simon were instrumental in the development of the theory and model. Additional advice came from many scientists at CMU, Xerox PARC, and IBM. Special thanks to Jeff Siskind, who assisted in the analysis of IE. The intelligibility of this paper is largely due to the efforts of Pat Langley and Jack Mostow.

References

Bott, R. A. (1979). *A study of complex learning: Theories and methodologies* (Technical Report No. 7901). San Diego: University of California, Center for Human Information Processing.

Bower, G., & Trabasso, T. (1963). Concept identification. In R. C. Atkinson (Ed.), *Studies in mathematical psychology.* Stanford, CA: Stanford University Press.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking.* New York: Wiley.

Burstein, M. H. (1986). Concept formation by incremental analogical reasoning and debugging. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Cullingford, R. (1981). SAM. In R. C. Schank & C. K. Riesbeck, *Inside computer understanding.* Hillsdale, NJ: Lawrence Erlbaum.

Darden, L. (1976). Reasoning in scientific change: Charles Darwin, Hugo de Vries, and the discovery of segregation. *Studies in the history and philosophy of science, 7,* 127–169.

Dietterich, T. G. (1984). *Constraint propagation techniques for theory-driven data interpretation.* Doctoral dissertation, Department of Computer Science, Stanford University, Stanford, CA.

Dietterich, T. G., & Buchanan, B. G. (1983). The role of experimentation in theory formation. *Proceedings of the Second International Machine Learning Workshop.* Urbana, IL.

Dietterich, T. G., & Michalski, R. S. (1985). Discovering patterns in sequences of events. *Artificial Intelligence, 25,* 187–232.

Gentner, D., & Gentner, D. R. (1983). Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner & A. L. Stevens (Eds.), *Mental models.* Hillsdale, NJ: Lawrence Erlbaum.

Greiner, R. (1985). *Learning by understanding analogies* (Technical Report No. STAN-CS-85-1701). Stanford, CA: Stanford University, Department of Computer Science.

Hammond, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 556–560). Philadelphia, PA: Morgan Kaufmann.

Hayes-Roth, F., Klahr, P., & Mostow, D. J. (1981). Advice taking and knowledge refinement: An iterative view of skill acquisition. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Erlbaum.

Huesmann, L. R., & Cheng, C. M. (1973). A theory for the induction of mathematical functions. *Psychological Review, 80,* 126–138.

Hunt, E. (1962). *Concept learning: An information processing problem.* New York: Wiley.

Kotovsky, K., & Simon, H. A. (1979). Empirical tests of a theory of human acquisition of concepts for sequential patterns. *Cognitive Psychology, 4*, 399–424.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). *Universal subgoaling and chunking.* Boston, MA: Kluwer Academic.

Langley, P., Simon, H. A., Bradshaw, G. B., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes.* Cambridge, MA: MIT Press.

Moran, T. P. (1981). The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies, 15*, 3–50.

Mynatt, C. R., Doherty, M. E., & Tweney, R. D. (1977). Confirmation bias in a simulated research environment: An experimental study of scientific inference. *Quarterly Journal of Experimental Psychology, 29*, 85–95.

Schank, R. C. (1983). *Dynamic memory.* New York: Cambridge Press.

Schank, R., & Abelson, R. (1977). *Scripts, plans, goals and understanding.* Hillsdale, NJ: Lawrence Erlbaum.

Shrager, J. (1985). *Instructionless learning: Discovery of the mental model of a complex device.* Doctoral dissertation, Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA.

Shrager, J., & Klahr, D. (1983). Learning in an instructionless environment: Observation and analysis. *Human Factors in Computing Systems* (pp. 226–229). Association for Computing Machinery: Boston, MA.

Shrager, J., & Klahr, D. (1986). Instructionless learning about a complex device: The paradigm and observations. *International Journal of Man-Machine Studies, 25*, 153–189.

Simon, H. A. (1979). The information storage system called "human memory." In H. A. Simon (Ed.), *Models of thought.* New Haven, CT: Yale University Press.

Wilensky, R. (1983). *Planning and understanding.* Reading, MA: Addison-Wesley.

Zytkow, J. M., & Simon, H. A. (1986). A theory of historical discovery: The construction of componential models. *Machine Learning, 1*, 107–137.