Situated Decision-Making and Recognition-Based Learning: Applying Symbolic Theories to Interactive Tasks

Alonso H. Vera
Department of Psychology

Richard L. Lewis
School of Computer Science

F. Javier Lerch
Graduate School of
Industrial Administration

Carnegie Mellon University
Pittsburgh, PA 15213
av0m@andrew.cmu.edu

Abstract

This paper describes two research projects that study typical Situated Action tasks using traditional cognitive science methodologies. The two tasks are decision making in a complex production environment and interaction with an Automated Teller Machine (ATM). Both tasks require that the decision maker and the user search for knowledge in the environment in order to execute their tasks. The goal of these projects is to investigate the interaction between internal knowledge and dependence on external cues in these kinds of tasks. We have used the classical expert-novice paradigm to study information search in the decision making task and cognitive modeling to predict the behavior of ATM users. The results of the first project strongly indicate that decision makers are forced to rely on environmental cues (knowledge in the environment) to make decisions, independently of their level of expertise. We also found that performance and information search are radically different between experts and novices. Our explanation is that prior experience in dynamic decision tasks improves performance by changing information search behavior instead of inducing superior decision heuristics. In the second study we describe a computer model, based on the Soar cognitive architecture, that learns part of the task of using an ATM machine. The task is performed using only the external cues available from the interface itself, and knowledge assumed of typical human users (e.g., how to read, how to push buttons). These projects suggest that tasks studied by Situated Action research pose interesting challenges for traditional symbolic theories. Extending symbolic theories to such tasks is an important step toward bridging these theoretical frameworks.

This research was sponsored in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597 and in part by the USPS, Contract Number 104230-91-H-3819.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the USPS or the U.S. government.

The three authors contributed equally to the writing of this paper. We thank Cathleen Wharton for drawing our attention to the ATM task (Wharton, 1991).

Introduction

The main goal of this paper is to illustrate the role of (internal) knowledge in Situated Action tasks. One of the major objections of Situated Action researchers is that traditional symbolic theories are incapable of explaining human behavior in a variety of real-life tasks in which information or knowledge is situated in the environment. We describe two research projects that study tasks that force problems solvers to find and utilize knowledge in the environment before actions are taken. Our primary assertion is that problem solvers in these situations use their internal knowledge to *search* for knowledge in the environment before executing the tasks.

The first project investigates decision making in a complex production environment. We have studied how first-line supervisors manage manpower and machine resources in a high volume mail sorting facility. These supervisors make daily decisions on the assignment of people and machines to different sorting tasks in an information rich environment. Most of their information can only be accessed by visually scanning the workfloor or verbally interacting with other workers in the production floor. The ultimate objective of the project is to design decision support tools to improve machine utilization and the meeting of service objectives. In this paper we report a selected set of results from the first two phases of this project. In the first phase we studied the supervisors' information search behavior on the workfloor by following them during their entire shift (10 hours) and recording their actions and interactions with other workers. In the second phase we built a computer animation tool that simulates the workfloor and we asked the supervisors to interact with this tool by making decisions similar to those in their daily work-life.

In both phases we used the classical expert-novice paradigm to contrast the information search behavior of experienced supervisors against novices. We assumed that, independently of their level of experience (or expertise), supervisors need to access information and knowledge from the environment since it is impossible to build a complete plan for the entire shift. Our theory prescribes that the supervisors use their prior knowledge to interpret the information on the workfloor and to decide if more information or knowledge from other employees is needed before a decision is made. We did not expect performance differences between experts and novices to be the result of experts having more sophisticated decisions heuristics. We did expect that more experienced supervisors have learned how to *interpret* environmental cues more efficiently and how to look for information more effectively. A more sophisticated information search behavior should result in improved performance because better information is available to make decisions.

The second project studies how to build a symbolic model of users learning to interact with ATMs. We assumed that most ATM users never develop complete plans of how to interact with the machines to accomplish a variety of tasks. We hypothesize that users, in general, read the available instructions and build sketchy representations of the device during their first interaction. In subsequent interactions, users are expected to take advantage of this internal knowledge to interpret the external cues (external knowledge) provided by the interface. We have built a model of these interactions within a traditional symbolic architecture. This model operates with incomplete knowledge of how to perform tasks, but it takes advantage of the knowledge embedded in the environment.

The symbolic model was implemented in Soar. The model first simulates the user reading the instructions provided by the interface and then builds a behavioral representation from them. For example, after having read the instructions for inserting the card, the model builds an internal representation indicating that you need to insert your card into a slot to execute ATM transactions. This representation will then be used to interact with the machine. However, what is ultimately learned from the interaction is not a complete and explicit plan, but rather a recognition memory of a piece of behavior. For example, the next time the same user wants to check her account balance, she may notice that there is a slot in the machine, use her general knowledge about slots (e.g., you insert cards in them), and then recognize that inserting the card is the first behavior required for performing tasks with these machines.

These two projects illustrate the basic working assumption of cognitive science that what is inside the head is a simple representation of the relevant aspects of a complex world (Simon, 1969). These studies suggest that humans build simple representations of complex and routine tasks that later allow them to take advantage of information in the environment. The two studies were conducted within the traditional symbolic framework in cognitive science while taking into account the concerns raised by Situated Action researchers. In the first project, we used an ethnographic method in the first phase of the project to understand the dynamics of the workfloor environment. The results of the first phase allowed us to construct an animation tool that incorporates fundamental and relevant aspects of the supervisors' job. With the help of this simulated environment, we exploited the classical expert-novice paradigm to explore how internal knowledge is used to access knowledge in a complex real-life environment. In the second project, Situated Action theory helped us to develop a symbolic model that does not require memorization of elements of the task. The model combines knowledge from the environment with simple internal knowledge about basic objects (e.g., slots, buttons, screens) and what to do with them.

Decision Making Study

This section describes the first two phases of the decision making project. This project has been conducted during a three year period. Before discussing the specifics of these two phases, it is useful to describe the production setting and the nature of decision making tasks.

Production Environment

The production facility is a large physical plant with approximately 3000 employees in a space the size of about six football fields. The facility processes about six million pieces of mail per day. Most mail is processed in the automation section which consists of two different processing areas: the optical character reader (OCR) area and the bar code sorter (BCS) area. OCRs read the address on envelopes and spray a bar code onto them. This mail is then sent to the BCS area where it is sorted to different levels of granularity according to its destination. A piece mail may therefore be processed anywhere from one to five times in the BCS area, depending on its type. Decisions in the BCS area are contingent on the activities of *upstream* areas such as the OCR area.

Our study has focussed mainly on how first-line supervisors search for information and make decisions in the BCS area. There are two main things that supervisors need to make decisions about: allocation of manpower to the BCS machines and the assignment of sorting programs to each machine. At the time of the study, there were approximately one hundred different sorting programs for the BCS machines. Different programs are needed to process mail for specific destinations. Supervisors need to continually assess the volume of different mail types in order to assign the appropriate sorting programs to meet dispatch times. This decision making task is complicated by the fact that there are interactions among sorting programs. That is, some programs only sort mail directly for dispatch while others sort mail that requires further processing by other programs. Therefore, the scheduling of sorting programs depends on the volume of different mail types, the availability of manpower, the sorting programs currently being run, the time of day, and the dispatch deadlines for each specific mail type.

The BCS environment lacks any computerized source of information. All the information has to be accessed by either visually scanning the workfloor or by directly asking other workers. Supervisors may ask for information from their subordinates in the BCS area or may use the telephone to call supervisors in other areas such as the OCR. Supervisors may also walk to the mail staging areas to see how much mail is waiting to be processed. The supervisors need to make their decisions in real time in a constantly changing environment. In sum, the facility is a typical Situated Action research setting.

Phase One

Our first goal was to understand the technology and the organizational setting in the BCS area. We spent 6 months learning the language used by supervisors and workers, the specific nature of the decision tasks, the content of the information exchanged among the actors, and the relationships between management and workforce. We also learned the technical details of the overall operation and how other units in the production facility interact with the BCS area.

After this initial familiarization process, we then develop a coding instrument for collecting detailed data about the activities and interactions of the BCS supervisors. Our goal was to have a detailed record of how supervisors spend their time, how they search for information, the content of this information, and the nature and content of their interactions. We tested the reliability of the instrument by calculating inter-observer reliability (see Lerch, Fenner, Snyder, and Goodman (1992b) for more details). In general we obtained a satisfactory level of reliability (Kappa values between .69 and .77 for the major categories). The coding scheme differentiates between information search activities (e.g., asking workers for information, looking for mail in the staging areas) and decision activities (e.g., assigning mail to machines, manpower selection and assignment).

We collected approximately 60 hours of shadowing data by following six supervisors with different levels of expertise for their entire shift (10 hours). We collected and coded 5570 interactions and activities for an average of 1.78 activities per minute. We used these data to test a set of eight information search hypotheses that compare information search behavior between experts and novices (See Lerch, Fenner, Snyder and Goodman, (1992a)). In general the results indicate substantial differences in information search behavior between experts and novices. We present below selected examples of these results to illustrate these differences.

Results. Experienced and novice supervisors spent approximately the same proportion of their time searching for information (71.7% vs. 71.9%), but experienced supervisors seem to be more efficient. For example, experienced supervisors spent 77% of their information search activities interacting with workers or other supervisors and only 23% of their time directly observing the workfloor. On the other hand, less experienced supervisors spent 66% of their information search time in interactions and 34% in direct observation. An analysis of the content of the interactions indicates that information about the environment is easier to obtain by asking others than by direct observation.

Experienced supervisors are also more effective in searching for prognostic information. For example, experienced supervisors search for information about upstream operations (e.g., OCR area) more than novices when interacting with other supervisors (26% vs. 13%). Upstream information is key for predicting future mail flows and for improving manpower planning and scheduling. The search for upstream information has to be initiated by the BCS supervisors since they need to call other areas using the telephone. The data suggests that experienced supervisors are more likely to search for upstream information because more situations seem to trigger the requirement of searching for information outside their immediate locus of control. For example, experienced BCS supervisors may be able to notice subtle changes in mail flows which may lead them to request upstream information.

In summary, experts actively search for more upstream information (e.g., OCR area) than novices. They also rely less on information directly accessible from the environment. Whereas novices tend to do more direct observation which gives them raw data about the environment, experts are more likely to ask other people for processed information.

Discussion. A situated explanation of these results might be that familiarity with the environment accounts for the differences in information search behavior between experts and novices. That is, differences between experts and novices may have little to do with knowledge inside their heads, but might instead be due to social and perceptual advantages gained by longer exposure to the environment. As a consequence of having been there longer, experts may simply have more friends and be familiar with more types of situations allowing them to get more affordances from their environments. Experts, by virtue of being more familiar with their environments, are able to abstract better information. Increasing acquaintance with one's surroundings allows one to make better use of external cues. Another potential situated interpretation of the results would be that the differences we found between novices and experts were due to their respective social roles and not to their experience. By observing them on the workfloor, we could not distinguish between the role of seniority and that of knowledge. Unfortunately, the results from Phase One do not allow evaluation of these two situated hypotheses against more symbolic alternatives (e.g.,that information search differences are due to differences in internal knowledge between experts and novices).

Although the methodology used in Phase One was very useful for understanding the nature of information search behavior, it is important to note that it also had significant disadvantages. Since we did not observe all the supervisors on the same day, it is possible that the days on which they were observed were different in important ways. Differences between experts and novices may therefore be due to differences in the tasks they faced.

As traditional cognitive scientists, we wanted to control the task faced by the supervisors (e.g., characteristics of mail volume and mail flows for different days) and to remove them from their social roles in order to minimize the advantage experts may have because of their more elaborate social networks. In order to address these two concerns, we proceeded to conduct more controlled experimentation in Phase Two.

Phase Two

In Phase Two, supervisors were asked to interact with an animated computer model of the BCS area. We took considerable care to faithfully reproduce typical situations. For example, the mail volumes simulated in the animation tool were based on those from a real day. We also conducted a set of pilot studies to evaluate the visual layout and interactional aspects of the simulation. We modified the tool as necessary to facilitate information search and interaction.

Supervisors searched for information on the animation tool by switching between different information screens. Supervisors could only look at one screen at a time. The animation tool ran on two different computer monitors: one monitor was dedicated to information presentation (with seven available screens) and the other was dedicated to input commands to assign sorting programs to machines. Supervisors were allowed to freeze simulated time to assign mail to machines or to browse for information. They were also allowed to browse for information while the simulation was running.

The animation tool simulates the mail flows and sorting activity during the night shift (from 10 pm to 6 am), which is referred to as Tour 1. Supervisors were grouped into three categories according to their experience. There were two groups of experts: Tour 1 experts were people who had been supervisors on the night-shift for three years or more. These subjects were therefore domain experts on the night shift task. The other group of experienced supervisors, Tour 3 experts, worked an earlier shift. They had three or more years of work experience on the day-shift, but no experience on the night shift. The third group was Tour 1 novices who worked the night shift and had been supervisors for less than 6 months. There were three supervisors in each group.

Each supervisor was run individually. The entire sessions was videotaped and verbal protocols were collected. The animation tool recorded and time-stamped all keystrokes. This allows us to measure how supervisors allocated their time searching for information in the different screens.

Results. The results are divided into two separate analyses: 1) comparison of Tour 1 experts and Tour 3 experts, and 2) comparison of Tour 1 experts and Tour 1 novices.

Tour 1 experts vs. Tour 3 experts. Performance was measured by the number of pieces of mail that were processed and sent out before their appropriate dispatch times. Since all supervisors faced the same situation, it is easier to report performance as the number of pieces of mail that failed to meet their dispatch times. There were no significant differences in the performance of Tour 1 experts and Tour 3 experts. On average Tour 1 experts missed 53,000 pieces of mail while Tour 3 experts missed 57,000.

Since we hypothesized that Tour 1 experts have better knowledge for running mail in Tour 1 when compared to Tour 3 experts, we analyzed their information search behavior to explain why performance differences were not found. Tour 3 experts overcame their lack of tour-specific knowledge by spending more time in information search. On average Tour 3 experts spent 25 minutes more in information search than Tour 1 experts (the average total time to perform the whole task was approximately one hour and a half). Tour 3 experts had a larger number of switches among screens (Tour 3: 236 screens vs. Tour 1: 180 screens), but the time spent in each screen was roughly the same (Tour 3: 28.1 sec vs. Tour 1: 28.4 sec.).

Table 1: **Percentage of time allocated by experts to city screen at two hour time intervals.**

	Tour 1 expert	Tour 3 expert
10:00 pm –12:00 am	4.9%	6.2%
12:00 am –2:00 am	1.9%	7.1%
2:00 am –4:00 am	21.4%	11.1%

Table 2: **Percentage of time allocated by Tour 1 supervisors to city screen at two hour time intervals.**

	Tour 1 expert	Tour 1 novice
10:00 pm –12:00 am	4.9%	4.8%
12:00 am –2:00 am	1.9%	1.9%
2:00 am –4:00 am	21.4%	16.0%

This suggests that both groups of experts are equally efficient in extracting information from the screens (i.e., same time per screen). However, Tour 3 experts perform a more exhaustive and systematic information search because they lack knowledge about when to look for the appropriate information. This hypothesis is supported by looking at the allocation of information search among screens during the course of the simulation. For example, Table 1 shows the percentage of time allocated to a specific screen (the “city mail” screen) at three different two-hour intervals. The data shows that Tour 1 experts allocate a very small percentage of time to the city screen between 10 pm and 2 am. This is because most of the action for city mail occurs after 2 am when Tour 1 experts begin to allocate a greater percentage of their information search to it. On the other hand, Tour 3 experts allocate almost twice the percentage allocated by Tour 1 experts between 10 pm and 2 am. In general, city mail information is mostly irrelevant for decision making before 2 am. Furthermore, Tour 3 experts allocate almost half of the percentage of Tour 1 experts to the city screen during the critical interval between 2 am and 4 am.

Tour 1 experts vs. Tour 1 novices. In contrast to the similar performance for the two groups of experts, we found significant performance differences between experts and novices from Tour 1. While experts missed only 53,000 pieces of mail, novices missed on average almost three times as much (153,000 pieces of mail). Both groups spent roughly the same length of time browsing, but novices had substantially fewer switches among screens (Experts: 180 screens vs. Novices: 130 screens). Therefore, novices spent significantly more time per screen (Experts: 28.4 sec. vs. Novices: 40.3 sec.). This suggests that novices are less efficient in extracting information from a single screen. On the other hand, Tour 1 novices follow a similar time allocation pattern among screens through the course of the simulation. For example, Table 2 shows the percentages of time allocated to the city screen for experts and novices. The data shows that both groups have similar percentages at each of the three time intervals. This indicates that although novices search for information less efficiently, they have the necessary tour-specific knowledge to mimic the global allocation pattern of Tour 1 experts.

Discussion. The comparison among the three groups of supervisors allows us to evaluate our proposed Situated Action interpretations of the results of Phase One. Since Tour 3 experts (i.e. supervisors with only day-shift experience) do not have the same degree of familiarity with the specific task characteristics as do Tour 1 experts, their performance should be worse. The finding that there is no performance difference between the two groups of experts may be construed as indicating that Tour 3 experts compensate for their familiarity deficiency by searching for more information. But this explanation is not supported by the fact that novices fail to compensate for their own familiarity deficiency in the same manner. An alternative explanation is that Tour 3 experts have enough internal knowledge about the general nature of the task so that they are able to utilize external information when available, even though they lack the specific knowledge to search for information in the same directed way as Tour 1 experts.

The second situated explanation proposed in Phase One was that experts had more sophisticated social networks that allow them to search for information more efficiently and effectively. This interpretation is contradicted by the results in Phase Two. Tour 1 experts can not take advantage of their social network when using the animation tool. Nevertheless they show more efficient information search than Tour 1 novices (e.g., time per screen) and more effective information search than Tour 3 experts (e.g., total browsing time, the allocation of information search among screens at different time intervals). Again, these results hint that Tour 1 experts possess more differentiated internal symbolic structures that guide them in their information search behavior.

The empirical results from the two phases show that decision making in dynamic environments consists of two separate processes: First, decision makers search and access information from the environment, and second, they process this information to make decisions. Situated Action research suggests that dynamic decision making environments provide sufficient information for the second process to be completed without significant problem solving or deliberation. Their argument is that once the first process is accomplished, the second follows easily in tasks where the environment provides sufficient cues. In general we agree with this characterization of decision making in dynamic environments but contend that internal knowledge plays a key role in determining the value of the information accessed during the first process. In other words, expert decision makers in these environments are better at finding the appropriate information at the right time, and they can find it faster than novices.

In order to make recommendations on how to design computer-based decision-support tools, we have carried out a detailed study of the daily activities of first-line automation supervisors. We are in the process of developing a prescriptive methodology intended to improve bad features of the current environment while maintaining positive affordances already present. Our research has focused on understanding the human processes underlying information search in dynamic decision making tasks in order to achieve this goal.

An alternative approach to studying this task would have been to build a computer model of a supervisor. Unlike chess or physics problems, however, the task facing supervisors is not well-defined. Moreover, at the start of the project our understanding of the task was not sufficient to allow us to confidently account for a significant proportion of the variables affecting supervisors' information search behavior or performance. Since the tasks and the role of the person in these environments are usually too complex, building models of human cognitive processes is likely to fail in providing significant insights. A more modest strategy for cognitive modeling is to select a simpler situated action task such as the interaction with single-function devices such photocopiers and ATMs. In the next section, we describe a cognitive model in Soar, a classic symbolic architecture, that simulates users learning and performing interactions with a simplified ATM interface.

A Soar model of an ATM user

There are a variety of common human-computer interaction tasks that, while being fairly self-explanatory, require the user to perform behaviors which are initially novel to them. Using a banking machine (ATM), a photocopier, a fax machine, and so on, all require the user to navigate through a series of responses and decisions which are not typically anticipated by the user. The first time a person uses a particular ATM, for example, they are likely to closely read and follow the given instructions. Through repeated use though, the person will usually learn to perform at least some parts of the task without referring to the instructions. Nevertheless, it is unlikely that ATM users can recall the instructions or the specific physical characteristics of the interface (see (Payne, 1991) on Macintosh users recall of interface characteristics).

The Soar model we describe proposes that learning in these tasks is interactive, and that the representation of the task remains dependent on perceptual cues provided by the environment. Our model learns to recognize patterns in the environment and to perform behavioral responses when presented with them. Nevertheless, the learning process does not result in a complete representation of the interface nor an explicit plan for how to complete the task.

Soar is a theory of the human cognitive architecture embodied in an implemented programming system (Laird, Newell and Rosenbloom, 1987; Newell, 1990). All behavior in Soar occurs in problem spaces, where operators are applied to states to make progress towards a goal. The current problem space context is represented in a short term working memory. All the knowledge that guides behavior (e.g., what operators to apply, how to apply them) is held in a long term recognition memory, which continually matches against the working memory. This recognition memory consists of a set of *condition-action associations*.

When no more progress towards the goal can be made in the current problem space, an impasse arises, signaling a lack of immediately available knowledge in the recognition memory. Soar responds to the impasse by generating a subgoal to acquire the missing knowledge in other problem spaces. These problem spaces can apply general problem-solving heuristics and task specific-knowledge until the problem is solved or another impasse occurs. When the impasse is resolved, Soar's learning mechanism builds new associations in LTM which capture the results of the subgoal. The next time Soar encounters a similar situation, the associations allow the system to recognize immediately what to do.

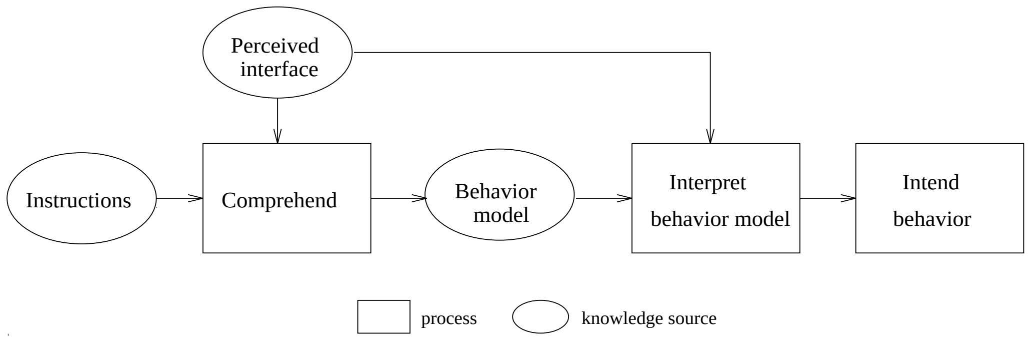

Figure 1: **Performing the task with instructions for the first time.**

The task

The ATM machine our model will use has a slot for inserting a bank card, an orange button, a green button, and an alphanumeric display for instructions. The task is retrieving a checking account balance. The machine will first display the instruction "INSERT CARD INTO SLOT", and once this step is successfully completed, it will display "PRESS THE ORANGE BUTTON FOR ACCOUNT BALANCE, GREEN BUTTON FOR DEPOSIT". Though this task and interface is an abstraction of a real situation, it still permits exploring the questions of interest: How does the model initially perform the task by following instructions? What kind of learning goes on as a result of instruction taking? How does the model subsequently perform without the instructions? What is the nature of the internal representations?

Behavior of the Soar model

Consider what must happen functionally for the user to succeed in performing the initial step of inserting the card. The user must first *comprehend* the natural language instructions, extracting from the sequence of words an internal representation of their meaning. Then the user must *operationalize* this information within the current context by generating an intention to actually perform the action of inserting the card (Mostow, 1981; Huffman and Laird, 1992). Finally, the user must successfully execute the intended motor behavior.

Figure 1 shows the sequence of events leading to the "insert" action. (This is a simplification of the actual set of problem spaces and operators evoked by the task; the process boxes in the figure do not represent separate modules to which control is passed.) The initial written instructions are comprehended by applying Soar's existing natural language capability (Lehman, Lewis and Newell, 1991; Lewis, 1993a). Comprehension of instructions produces a temporary declarative meaning representation in working memory which we will call a *behavior model*. In this case, it represents the action of inserting the card. The behavior model is not itself the behavior though; Soar must transform this declarative representation into executable operators (Lewis, Newell and Polk, 1989).

Since Soar has had no prior experience with this task, no such operators are directly available and an impasse arises. Soar then enters another problem space where it tries to interpret the behavior model and produce a new operator directly executable in the current context. It is this process of *interpretively executing* the behavior model that results in the operationalization of the instructions. The mapping from behavior model to action is relatively simple in this case, though in general it can be complex (Huffman and Laird, 1992). The executable operator which results is the *intention* to perform the relevant motor actions that will result in the insertion of the card. As a result of resolving this impasse, Soar stores an association in LTM of the form:

```
IF there is a behavior model object encoding
      "insert the card into the slot"
   and there is a card and a slot
   and the goal is to retrieve the checking balance
THEN propose the operator that intends the
   motor actions to insert the card
```

Figure 2 shows what happens when the same task is performed again with the instructions. Soar comprehends the instructions as before. Once the behavior model is produced, the learned association immediately proposes the intention to insert the card, thereby avoiding going into the interpretation problem space. Thus, the effect that learning has is to convert the operationalization of the instructions from a deliberate process to one of recognition.

The only memory of the task that Soar carries from one ATM episode to the next consists of the association described above that arises during task performance. What happens if the instructions are absent, or ignored? The association that proposes the insert action will not be

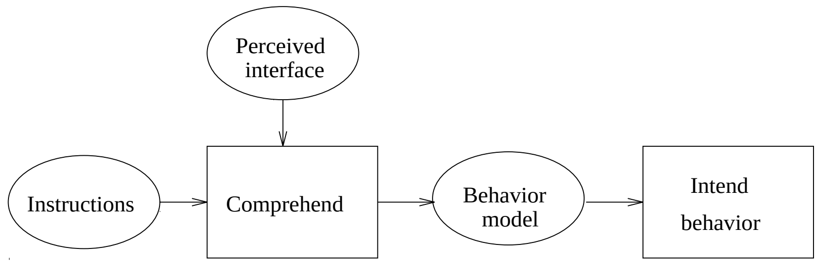

Figure 2: **Performing the task again with instructions.**

evoked since its conditions test for the behavior model that resulted from comprehending the instructions. It would appear that Soar cannot re-execute the task without explicit instructions. Soar can *recognize* the correct action to take, but cannot *recall* it. The only way to evoke the association is to assemble the appropriate cues in working memory by generating behavior model representations of candidate actions. In the ATM task, the obvious generator is the interface itself, along with general knowledge about slots and buttons.

Figure 3 shows the processes that arise when Soar attempts to perform the task without the instructions. Soar engages in a *generate-and-test* behavior which proceeds as follows. Without instructions, an impasse arises since no comprehension or intention operators are immediately available. Soar enters a space for *guessing* the appropriate action to take. This space of possible actions is constrained by the external cues present in the interface. These cues trigger associations that propose Guess operators. Knowledge about affordances in the environment is thus encoded in the proposals for these guesses. Since there is a button on the interface, a guess is proposed and applied that creates a behavior model object representing a *push action*. Nothing happens as a result of guessing the push action, so another guess, triggered by the presence of the slot, creates an object representing an *insert action*. Since this object has the same structure as the behavior model built from the instruction the first time the task was performed, it immediately evokes the association to propose the operator that intends the insert behavior. Soar now remembers that this was the action it took in the first episode.

The proposal of the intend operator resolves the initial impasse, and Soar builds a new association in LTM:

```
IF there is a card and there is a slot
    and the goal is to retrieve the checking balance
THEN propose the operator that intends the
    motor actions to insert the card
```

Since the behavior model was not part of the pre-impasse context (because the instructions were absent), the resulting association does not test for it. Figure 4 shows what happens when Soar performs the task on subsequent trials: this association will be evoked immediately, proposing the intention of the insert action. There is now a direct mapping from the cues in the environment to the desired behavior.

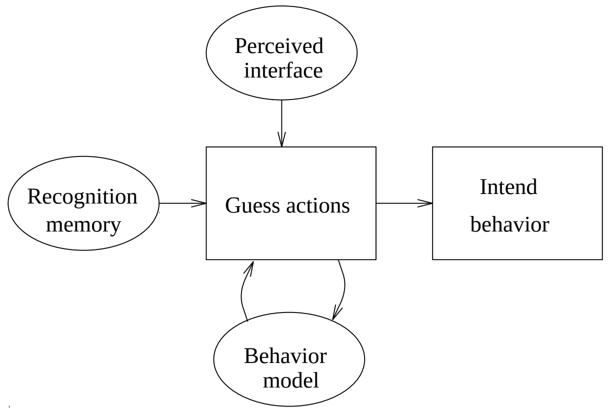

Figure 3: **Performing the task without instructions.**

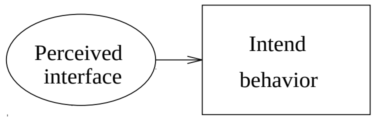

Figure 4: **Performing the task again without instructions.**

Theoretical implications

While extremely simple, this model has several interesting characteristics. The initial execution of the task requires some deliberate interpretation processes to operationalize the instructions. This interpretive process improves with learning. The memory of the task and instructions that results is fine-grained and recognitional—a complete, declarative plan is never stored away. Because the task memory is recognitional, retrieving the content of the instructions is a reconstructive process. The reconstruction is driven by a combination of the immediate environment and knowledge about general affordances.

These affordances are not just a programming hack; they fill a role functionally required by the way learning works in Soar. The most simple imaginable model in Soar yields associations that demand the kind of knowledge captured by affordances[1]. The final result of learning is a set of associations that test specific aspects of the interface and goal context, and propose intention operators. This results in reactive behavior with no intermediate declarative representations.

All of this derives from building a Soar model with a couple of simple assumptions about the user: 1) The user initially performs just what is functionally required to carry out the task; and 2) The user has a body of general knowledge about things like buttons and slots that enable him or her to generate possible actions for a given interface. The nature of the learning, the memory of the task, the necessity for reconstruction, and the subsequent reactivity all arise directly from independent assumptions of the Soar architecture.

The model can be extended in interesting ways to explore different assumptions about the user. For instance, the user could adopt a strategy of *deliberate memorization* initially, which would lead to both a recognition and recall memory of the content of the instructions. Such preparation would eliminate the later cost of reconstructive recall. However, the users must make a deliberate decision to do this; it is not functionally required to perform the task the first time through. In general, a mix of behavior is possible, by memorizing certain parts of the task and reconstructing other parts. But even when the strategy involves memorization, learning will eventually result in reactive task performance.

The performance of this model can be described as learning to use instructions to *interpret* the external environment. The model progresses from using instructions to directly using the cues in the environment in order to achieve its goal. Shrager and Callanan (1991) present an account of how children use collaborative activity as a learning resource which is, in important ways, similar to the behavior of this Soar model. They observed child-parent diads baking muffins wherein the language of the interaction facilitates children's development of, not just cooking skills, but also active interpretation of the activity. The child learns how to use active language (i.e., instructions) to *interpret* the situation. Similar to our model, the process of understanding the instructions is only the first step. Learning how to become an active interpreter of the environment (without instructions) is the crucial step. The Soar model lends support to the view posited by Shrager and Callanan, and, in addition, presents a plausible cognitive mechanism that might underlie learning in these kinds of tasks.

[1]This is a specific case of the general data-chunking problem in Soar (Newell, 1990)

Not memorizing things when we do not have to is a very adaptive trait. This also keeps us in a "nice balance between stimulus-bound activity and stimulus-independent activity" (Newell and Simon, 1972, p. 805). Two clear predictions of this theory are that people will not be able to recall all the steps of the task (Payne, 1991), and increasing the number of beneficial affordances reduces the overall complexity of learning the task. Correspondingly, conflicting affordances should increase the need to memorize those parts of the task that are repeatedly stumbled on. For example, if there are a number of slots in the interface, each of which affords inserting, then explicit memorization of this aspect of the interface could be adaptive.

Soar and Situated Action

Now that we have seen how Soar behaves in a simple situated task, it is interesting to step back and consider Soar generally in terms of the concerns of situated cognition. To do this, we will evaluate Soar along several key dimensions proposed by Maes (1992). These dimensions putatively distinguish Behavior-Based AI (i.e., AI based on Situated Action principles) from Knowledge-Based AI (based on symbolic theories of cognition).

Integrated competences vs. single competence. As opposed to building systems with a single expertise, behavior-based AI proposes constructing single systems with a variety of integrated competences. Soar shares this goal, which is a central tenet of the general research paradigm of building integrated intelligent architectures (Newell, 1990; Laird, 1991)[2]. For example, Soar systems have been created that combine natural language instruction taking with other tasks (Lewis et al., 1989; Huffman and Laird, 1992). (These competences are relatively high level compared to those of behavior-based AI which have typically been motor and perceptual). Nevertheless, we are still far from integrating all the individual Soar systems into a single agent—integration continues to be an important area of research for the Soar community.

Situated/open systems vs. non-situated/closed systems. Behavior-based AI focuses on constructing systems directly situated in the environment, which must operate in real time and be interruptable. Real-time operation and interruptability are important features of Soar as an AI architecture (Pearson, Huffman, Willis, Laird

[2]Behavior-based AI focuses on systems with independent competence modules, which appears to contrast with Soar's approach of finding general mechanisms that cover a range of domains. For an interesting discussion on modularity (Fodor, 1983) and Soar, see (Newell, 1990) and (Lewis, 1993b).

and Jones, 1993) and as a theory of cognition (Newell, 1990). In fact, the real-time constraint is the most fundamental constraint shaping cognitive theories within Soar. For example, the *immediacy of interpretation* constraint of language comprehension—that language is rapidly and incrementally comprehended on a word by word basis—is the primary factor shaping the Soar theory of language comprehension. Soar must be able to recognitionally bring to bear multiple knowledge sources to process utterances syntactically, semantically, and referentially (Lehman et al., 1991; Lewis, 1993a, this volume).

Behavior-producing structures vs. declarative/static structures. Behavior-based AI focuses on systems with structure that is primarily active, as opposed to static, declarative representations that must be interpreted to yield behavior. Nearly all structure in Soar consists of a massively parallel, recognition memory which is neither static nor declarative. No interpreter examines the associations in recognition memory: they are active processes evoked whenever their relevant cues are present in working memory. Only Soar's working memory contains what may be characterized as static and declarative representations.

Developing systems vs. non-developing systems. Behavior-based AI focuses on systems that acquire their behavior-producing structure and evolve with experience, as opposed to systems in which the structure is simply posited without concern for how it arose. This concern for learning is also a critically central issue for Soar as an AI system and as a cognitive theory. Soar continually learns through the experience-based chunking mechanism. For every piece of programmed structure in Soar, the question naturally arises: from what set of tasks and problem spaces could this structure have arisen by learning? Every programmed association in Soar must ultimately be held accountable in terms of how it might be learned, or why it might plausibly be innate.

Another issued raised by Maes is the nature of what is learned: does the system just compile what it already knows, or does it learn new things from the environment? Acquiring new knowledge has been an important research area in Soar (Rosenbloom, Newell and Laird, 1991) since a concern arose early on that chunking could only compile existing knowledge. Over the past few years, several Soar systems have been developed that do learn new things from the external environment. The simple ATM system described above is a good example. It starts out with no knowledge of the initial step to take in operating the machine; it acquires that knowledge from the external instructions.

Soar is even being applied to difficult problems in uncovering the mechanisms of cognitive development. Q-Soar is a system that models three and four-year old children learning some aspects of number conservation knowledge (Simon, Klahr and Newell, 1992). The model learns by participating in (a simulation of) an experimental training study used to train and test conservation.

Emergent vs. planned activity. In Behavior-based AI systems, activity is an emergent property of interacting modules and the immediate environment. There is no plan structure that is deliberated upon and then executed. While it is true that Soar can deliberate and plan (thereby correctly predicting that humans can as well), there is no single structure in Soar that corresponds to the plan. Rather, the knowledge that prepares Soar for behavior is distributed among independent associations in the recognition memory. Furthermore, there is no execution phase. Rather, planning and acting may be interleaved (Laird, Yager, Hucka and Tuck, 1990). The behavior that occurs in any particular episode emerges from an interaction of the knowledge in the recognition memory with the demands of the current task. Novel behavior may arise in a given situation because that situation evoked associations that never before had an opportunity to interact. This is all possible because Soar's control structure is *open*: no single module, problem space, subprocedure or production may ever gain complete control (Newell, 1973). Thus, the behavior does not correspond to the execution of a rigid plan structure. Admittedly, it is possible to build a Soar system that commits itself to a sequence of actions and is not interruptible by changes in the environment. But this is not an architectural constraint. Understanding how best to build Soar systems that respond flexibly in a dynamic environment is an area of active research.

Situated Action and Behavior-Based AI raise important issues that any theory of human or artificial intelligence must address. Nevertheless, after evaluating Soar along these dimensions proposed by Maes, we believe that Soar is an example of a symbolic cognitive architecture and research program that shares many of the concerns of the Behavior-Based approach.

General Discussion

We have presented two studies of very different situated tasks that involve extracting knowledge from the environment. The first study examined the decision-making behavior of novices and experts in a dynamic and complex mail sorting facility. Phase One of the study followed supervisors on their daily routines and showed that novices and experts differ in their search for the information they use to make decisions. These differences included the frequency with which novices and experts consulted other workers for the necessary information. The animation tool used in Phase Two controlled for the effects of familiarity and social interaction, and revealed that the differences in the information search behavior must be attributed at least in part to differences in knowledge alone.

With the Soar model of the ATM task, we explored in a simplified domain the *mechanisms* that might underlie acquiring knowledge from the external world—in this case, from instructions. We described how the Soar model makes a transition from behavior that results from deliberation about the instructions to behavior that is directly triggered by elements of the interface, without intermediate declarative representations. By assuming that subjects only do what is minimally necessary to perform the task each time, we discovered that the model actually predicts a dependence on cues in the external environment throughout the learning process. Such a model demonstrates the potential explanatory power of architectural theories in situated tasks.

These studies demonstrate the effectiveness of developing a complete approach to studying behavior in situated tasks. A common technique of Situated Action research has been to do very detailed analyses of tasks such as photocopying (Agre and Shrager, 1990; Suchman, 1987). If the goal of the research is to understand photocopying as an *activity*, then the focus should appropriately be on the activity itself. People should only be studied as interactive components of the activity. On the other hand, it is clear that in each of these activities, the human participant does have independent mental activity and that this mental activity is important to understanding why humans behave the way they do. If we want to understand the *person* engaged in these activities, then part of what we must understand is how that person acquires and uses knowledge in the activity. This requires understanding how behavior is modulated by knowledge (as in the decision making study) and the nature of the mechanisms used in the acquisition and application of the knowledge (as in the Soar study).

Thus, the challenge now is to continue to integrate what is being learned from symbolic cognitive science and Situated Action research to further our understanding of human cognition. One way to create bridges between the two approaches is to apply them to each other's traditional tasks. Our goal, as traditional cognitive scientists, is therefore to construct symbolic models that help us understand performance and behavior in situated tasks. Some situated action researchers may correspondingly take it as their job to provide an account of how a situated or behavior-based system might play chess or learn to solve Tower of Hanoi problems.

References

Agre, P. E. and Shrager, J. (1990). Routine evolution as the microgenetic basis of skill acquisition. In *Proceedings of the 12th Annual Meeting of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Fodor, J. A. (1983). *Modularity of Mind: An essay on faculty psychology*. MIT Press, Cambridge, MA.

Huffman and Laird (1992). Dimensions of complexity in learning from interactive instruction. In Erikson, J., editor, *Proceedings of Cooperative Intelligent Robotics in Space III, SPIE Volume 1829*.

Laird (1991). Special section on integrated cognitive architectures. *SIGART Bulletin*, 2(4):12–184. Papers presented at the 1991 AAAI Spring Symposium on Integrated Intelligent Architectures.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

Laird, J. E., Yager, E. S., Hucka, M., and Tuck, C. M. (1991). Robo-soar: An integration of external interaction, planning, and learning using soar. In de Velde, V., editor, *Machine Learning for Autonomous Agents*. MIT Press: Bradford Books, Cambridge, Massachusetts.

Lehman, J. F., Lewis, R. L., and Newell, A. (1991). Integrating knowledge sources in language comprehension. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 461–466.

Lerch, F. J., Fenner, D. B., Snyder, P., and Goodman, P. S. (1992a). Information search in real-time dynamic decision making. In *Proceedings of 5th Advanced Technology Conference*, volume 3, pages 893–912.

Lerch, F. J., Fenner, D. B., Snyder, P., and Goodman, P. S. (1992b). Managerial decision making in the automation section. USPS Center of Excellence on Information Technology and Decision Making, Carnegie Mellon University, working paper.

Lewis, R. L. (1993a). An architecturally-based theory of sentence comprehension. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*. In preparation, to appear.

Lewis, R. L. (1993b). Architecture Matters: What Soar has to say about modularity. In Steier, D. and Mitchell, T., editors, *Mind Matters: Contributions to Cognitive and Computer Science in Honor of Allen Newell*. Erlbaum, Hillsdale, NJ. To appear.

Lewis, R. L., Newell, A., and Polk, T. A. (1989). Toward a Soar theory of taking instructions for immediate reasoning tasks. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 514–521.

Maes, P. (1992). Behavior-based artificial intelligence. In *Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, Cambridge, MA. MIT Press.

Mostow, D. J. (1981). *Mechanical transformation of task heuristics into operational proceures*. PhD thesis, Carnegie Mellon University. Ph.D. thesis.

Newell, A. (1973). Production systems: models of control structures. In Chase, W. G., editor, *Visual Information Processing*. Academic Press, New York.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts.

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

Payne, S. J. (1991). Display-based action at the user interface. *International Journal of Man-Machine Studies*, 35:275–289.

Pearson, D. J., Huffman, S. B., Willis, M. B., Laird, J. E., and Jones, R. M. (1993). Intelligent multi-level control in a highly reactive domain. In *Proceedings of the International Conference on Intelligent Autonomous Systems*. To appear.

Rosenbloom, P. S., Newell, A., and Laird, J. E. (1991). Towards the knowledge level in Soar: The role of the architecture in the use of knowledge. In VanLehn, K., editor, *Architectures for Intelligence*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Shrager, J. and Callanan, M. (1991). Active language in the collaborative development of cooking skill. In *Proceedings of the 13th Annual Meeting of the Cognitve Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Simon, H. A. (1969). *The Sciences of the Artificial*. MIT Press, Cambridge, MA.

Simon, T., Klahr, D., and Newell, A. (1992). The role of measurement in the construction of conservation knowledge. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 66–71, Hillsdale, NJ. Lawrence Erlbaum.

Suchman, L. A. (1987). *Plans and Situated Action: The Problem of Human Machine Communication*. Cambridge University Press, New York.

Wharton, C. (1991). Implications of the differences between cognitive architectures for human-computer interaction: A comparative study of Soar and the Construction-Integration model. Presented at CHI 1991 Doctoral Consortium, and CHI 1991 Short Talks Session, New Orleans, Louisiana, April 28–May 2, 1991.