Knowledge-directed Adaptation in Multi-level Agents

John E. Laird and Douglas J. Pearson
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109
laird@umich.edu, dpearson@umich.edu
FAX: (313) 747-1761

Scott B. Huffman
Price Waterhouse Technology Center
68 Willow Road
Menlo Park, CA 94025
huffman@tc.pw.com

Abstract

Most work on adaptive agents have a simple, single layer architecture. However, most agent architectures support three levels of knowledge and control: a reflex level for reactive responses, a deliberate level for goal-driven behavior, and a reflective layer for deliberate planning and problem decomposition. In this paper we explore agents implemented in Soar that behave and learn at the deliberate and reflective levels. These levels enhance not only behavior, but also adaptation. The agents use a combination of analytic and empirical learning, drawing from a variety of sources of knowledge to adapt to their environment.

Introduction

Over the last ten years, there has been a convergence in the design of architectures for constructing intelligent autonomous agents that must behave in complex environments. Many architectures support three levels of knowledge with corresponding levels of control: a reflex level made up of independent rules or a finite state machine, a deliberate level that provides simple decision making and the ability to pursue multiple goals, and a reflective layer for deliberate planning. In theory, each of these levels can draw from a variety of sources of knowledge to adapt to their environment. However, the vast majority of work on adaptive agents has emphasized only a single source of knowledge, reinforcement, with improvement only at a single reflex layer. The advantage of these approaches is that they can make use of simple feedback that is available in many environments; however, in return, agents using these techniques are limited in the complexity of behavior that they can learn.

In order to extend adaptivity to more complex agents — agents with multiple levels of knowledge and control — we have developed a more deliberate approach which has been instantiated within two systems built within the Soar architecture: Instructo-Soar (Huffman & Laird 1994; Huffman 1994) and IMPROV (Pearson & Laird 1996). We find that agents with multiple levels of knowledge and control are not only more able to achieve complex goals, they also can use these layers to adapt to their environment. These agents use analytic and empirical learning techniques to draw on multiple sources of knowledge, including external instruction, internal domain theories, and past behavior.

We have focused on systems that learn at the two higher levels, deliberate and reflective, within agents that employ all three levels of control. This research complements the work that has been done to combine empirical and analytic learning methods (such as EBNN (Mitchell & Thrun 1993), EBC (DeJong 1995) and EBRL (Dietterich & Flann 1995)) that typically focus on agents that use only a single level of control.

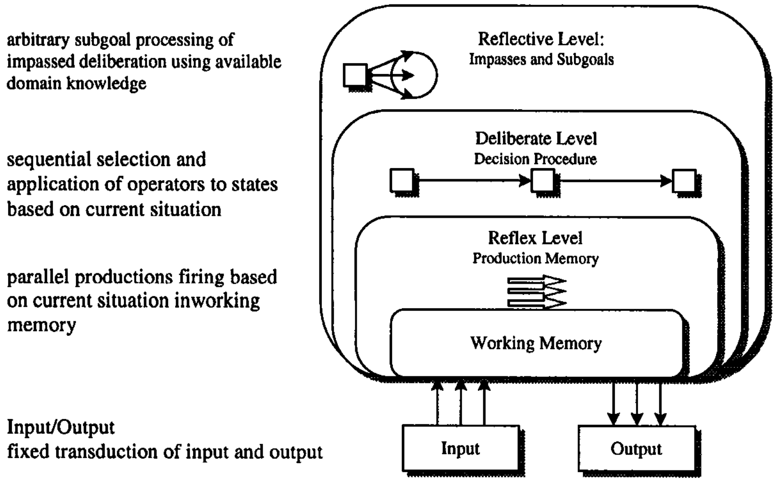

arbitrary subgoal processing of impassed deliberation using available domain knowledge
Reflective Level:
Impasses and Subgoals
sequential selection and application of operators to states based on current situation
Deliberate Level
Decision Procedure
parallel productions firing based on current situation inworking memory
Reflex Level
Production Memory
Working Memory
Input/Output
fixed transduction of input and output
Input
Output

Figure 1: Levels of Processing

Although our approach is deliberate in many ways, it is distinguished from deliberate approaches that treat the knowledge of the agent as a declarative structure that can be examined and modified at will. Such first-order declarative access to the agent's complete knowledge base poses significant computational complexity problems as the agent's knowledge grows in size and complexity. In contrast to our approach, these declarative/deliberate methods are therefore typically restricted to noise-free domains with instantaneous actions (Gil 1991; Ourston & Mooney 1990) which require relatively small domain theories.

In the remainder of this paper, we present our general approach to agent design and adaptation within the context of Soar. We then illustrate this approach by examining two integrated systems, Instructo-soar which learns to extend its knowledge through situated instruction, and IMPROV which learns to correct errors in its knowledge through interaction with its environment. Instructo-Soar uses analytic learning at the reflective level to gain new task knowledge at the deliberate level. IMPROV then uses largely empirical learning at the reflective and deliberate levels to identify and change incorrect knowledge learned from either bad instructions or incorrect generalizations of good instructions. IMPROV's recovery method can in turn be directed and guided by knowledge gained from further instructions processed by Instructo-Soar. We have combined IMPROV and Instructo-Soar to produce a single integrated system. This integration allows knowledge from instructions and from external experiences to be combined to improve the agent's learning and task performance.

Multi-Level Agent Structure

Our approach assumes that an agent consists of the three levels as shown in Figure 1.

1. The reflex level is where knowledge is directly and automatically retrieved based on the current situation. Most adaptive agents have only this level with the knowledge encoded in rules, finite-state automata, or similar fixed network structures. In Soar, the knowledge for this level is encoded in rules that fire in parallel whenever their conditions are matched.

2. The deliberate level is where knowledge is used to decide what to do next based on the current situation and goal. This allows for the integration of multiple chains of reasoning into a single decision in a manner that is not possible at the reflex level. In Soar, this level consists of the selection and application of deliberate operators. Rules propose and compare operators, and then a fixed decision procedure selects the preferred operator. Once an operator is selected, rules perform the appropriate actions. The advantages in terms of adaptation of this additional level of deliberations:

   (a) Selection knowledge (when and why to do something) can be learned and corrected independently of implementation knowledge (what to do).

   (b) Learning can be incremental, so that existing knowledge does not have to be modified (Laird 1988). This greatly simplifies learning so that it is not necessary to identify which piece of knowledge is incorrect, only which operator selection or implementation is incorrect.

   (c) Conflicts or gaps in the agent's knowledge can be detected when there is insufficient knowledge to decide which operator to select or apply next.

3. The reflective level is where arbitrary knowledge is used to either select the next deliberate action, or decompose a complex action into smaller parts. This level is invoked automatically when the deliberate level fails, either because there is insufficient knowledge to select or apply an operator. If there is insufficient selection knowledge, the agent can attempt to plan at the reflective level, using whatever knowledge it has of the domain to predict the worth of prospective actions. If there is insufficient application knowledge, the agent can treat the current action as a goal to be achieved, and then attempt to select and apply deliberate actions that can achieve that new goal.

   The advantages in terms of adaptation of this additional level of reflections:

   (a) The agent can create hypothetical situations distinct from its current sensing that it can reason about and learn from.

   (b) The agent can decompose a problem into subproblems that have commonalities with other, previously solved problems.

   (c) The agent can incorporate more declarative forms of knowledge, such as instruction.

   (d) Complex, time-consuming behavior at the reflective level can be compiled into rules for real-time processing at the deliberate and reflex levels.

Adaptation

In our approach, adaptation is based on a cycle in which the agent attempts to first correct or extend its reflective knowledge (which is essentially a multi-level domain theory), and then use the reflective knowledge as a basis for correcting its deliberate and reflex knowledge (which in turn influences which goals it can achieve in the world and can lead to the need for more extensions and corrections to its reflective knowledge, completing the cycle).

Figure 2 shows a simplified view of this cycle. In the first phase, information from the environment flows into the reflex level where it is parsed and interpreted. In parallel, the deliberate level is trying to make decisions so that the agent can pursue its goals. If the agent

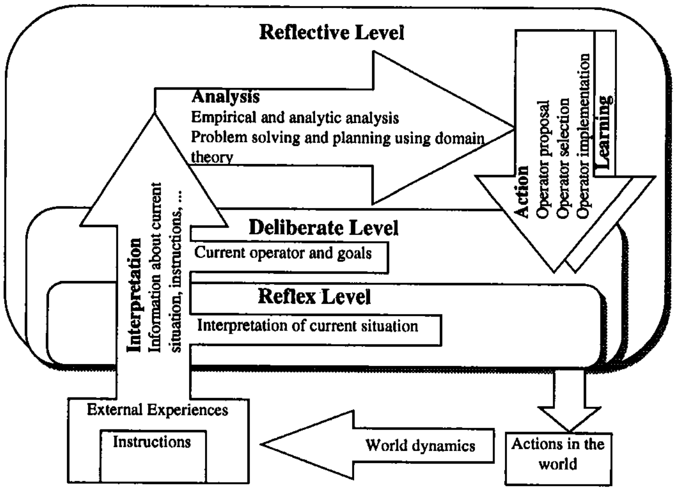


Figure 2: Adaptation of Multi-Level Soar Agent

is unable to make a decision this means its deliberate and reflex knowledge is in some way incomplete or incorrect. When this happens, the reflective level is automatically invoked.

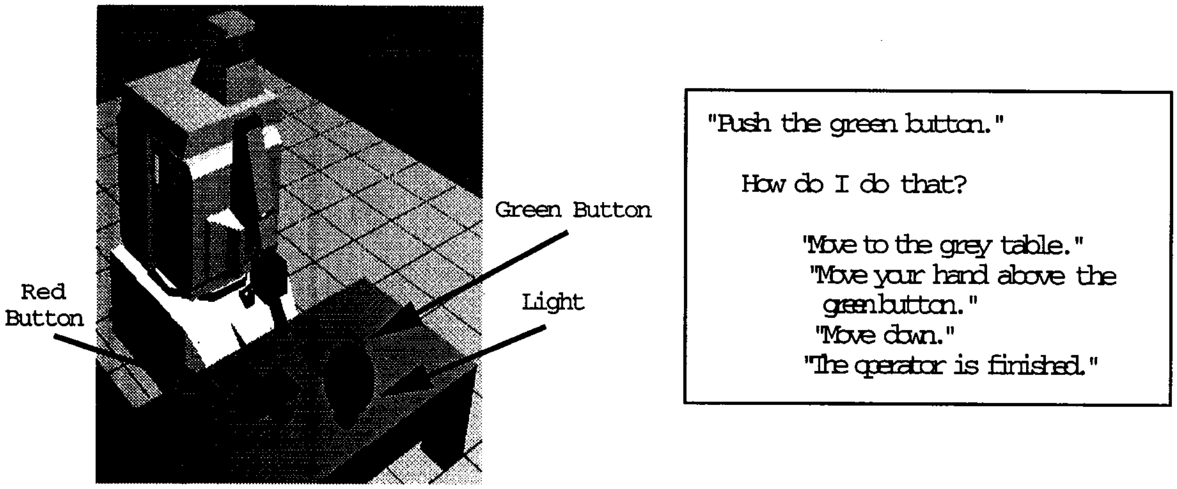
At the reflective level, the agent attempts to determine what part of the domain theory is either missing or incorrect. To date, we have focused on knowledge errors at the deliberate or reflective levels. If knowledge is simply lacking then instruction might be appropriate. However, if its experiences in the world suggest that its reflective knowledge is incorrect, then it uses both analytic and empirical means to determine how to correct its reflective domain theory. For some errors, this can be a complex process that involves multiple interactions with the environment that are not depicted in the figure. Once the domain theory knowledge has been corrected, it then uses it to replan its behavior. The results it produces are then compiled into knowledge that can be directly applied at the deliberate level in similar situations in the future.

Throughout the rest of the paper, we use a running example to describe the integrated instruction/recovery agent's performance. To date, we have only provided the integrated agent with enough domain knowledge for these simple examples, with the intention of proving the feasibility of the integration. First, we describe an instruction scenario that introduces errors into the agent's knowledge at the deliberate and reflective levels. Next, we discuss IMPROV's recovery technique applied to the error, first with no instruction available, and then with instruction directing each stage of the recovery process in turn.

Instruction-based Adaptation

In this section we give a review of Instructo-Soar which uses a specialization of the basic technique called *situated explanation* to extend its domain theory based on instruction. Instructo-Soar learns new procedures and extensions of procedures for novel situations, and other domain knowledge such as control knowledge, knowledge of operators' effects, and state inferences.

Instructo-Soar requests instruction whenever its deliberate knowledge is insufficient to determine which action to take next. The request for instruction is situated within its current



"Push the green button."

How do I do that?

"Move to the grey table."
"Move your hand above the greenbutton."
"Move down."
"The operator is finished."

Figure 3: (a) The example domain and (b) Instructions on how to push a button

task, and all Instructo-Soar asks for is what step should it take next. For each instruction, Instructo-Soar first determines what situation (state and goal) the instruction applies to – either the current situation, or a hypothetical one specified in the language of the instruction. Then, the agent attempts to explain at the reflective level, using its existing domain knowledge and forward projection, why the instruction will lead to success in the situation. If this explanation succeeds, the agent can learn general knowledge from the instruction (as in standard EBL (Mitchell, Keller, & Kedar-Cabelli 1986)). If the explanation fails, it means the agent is missing some knowledge required to complete the explanation. The missing knowledge can be acquired either through further instruction, or in some cases through simple induction over the "gap" in the incomplete explanation. However, either instructions or the agent's inductions can be incorrect and lead to learning errorful knowledge.

Instructo-Soar's domain includes a table with red and green buttons and a light on it (see Figure 3 (a)). The red button turns the light on and off, but the agent does not know this. Consider an example in which the agent is first taught a general procedure for how to push buttons, using the instructions in Figure 3 (b). Some time later, the instructor says:

"To turn on the light, push the red button."

To perform a situated explanation, the agent first determines that this instruction applies to a situation where the goal is to turn on the light. Then, using its existing domain theory, it forward projects the action of pushing the red button in that situation. However, since it does not know that the button affects the light, this projection does not explain how pushing the button causes the goal of turning on the light to be reached.

In this case, the agent makes a simple inductive leap to complete the explanation. It guesses that since there is a single action specified to reach a goal, that action directly causes the goal to be reached – e.g., pushing the button has the effect of turning on the light. Its inductive bias specifies that in this type of induction, the types of objects involved and their relationship to one another are important but other features, like the button's color, are not. Thus, the agent learns a rule that says "pushing any button on the same table as a light causes the light to turn on." This rule allows the agent to complete its explanation of the original instruction, producing a control rule that applies at the deliberate level that says "if the goal is to turn on a light on a table, choose the action of pushing some button on that table."

Previous versions of Instructo-Soar would ask the instructor to verify the inductive leap – Instructo-Soar was a "paranoid" agent, afraid to learn any wrong knowledge. However, as we will see, the addition of IMPROV, allows this verification step to be skipped, with the result being that errors in future performance will be corrected through interaction with the environment.

IMPROV's recovery technique

We will first describe each stage of IMPROV's recovery process and then how it can be informed by instruction. There are three principle stages to error recovery: (1) Detecting an error, (2) Finding a plan that, when executed, leads from the errorful state to achievement of the current goal, and (3) Identifying the piece of knowledge which led to the error and correcting it. IMPROV achieves these stages by (1) recognizing when the agent is no longer making progress in its problem solving, (2) using case knowledge of previous external interactions to guide a search for a successful plan, and (3) comparing successful and incorrect plans to identify the incorrect operators and then training an inductive learner on the results of executing the plans, which in turn leads to new case knowledge.

In our example, when the agent is asked to turn on the light, its overgeneral knowledge ("pushing any button causes the light to turn on") may lead it to push the wrong (green) button. When it does, IMPROV detects the failure (the light doesn't come on), searches for the correct plan (pushing the red button) and then learns that to turn on the light, the agent must push the red button, not the green one. Alternatively, at any (or all) stages of the recovery process, the instructor can provide simple guidance that speeds up the process by avoiding search.

Error Detection

IMPROV detects errors when executing a plan by recognizing that the agent is unable to make progress towards its goals. This can be because either it does not know how to achieve a goal from the current situation, or because it has conflicting suggestions as to what to do next. In our example, IMPROV detects an error when no operator is proposed after the agent pushes wrong button and the light doesn't come on. This is a weak error detection strategy that may signal errors when the agent's knowledge is correct but an unplanned for event occurs. However, reporting unplanned events as errors is acceptable as the agent's first step in recovering from an error is to replan.

Finding a plan that reaches the goal

After detecting an error in its deliberate knowledge, IMPROV searches and extends its reflective knowledge in an effort to correct the deeper reasoning that lead to the original error. It does this by searching for alternative plans, executing each in turn, to find one that satisfies the current goal [1]. This search is biased towards plans which are believed to be most likely to succeed, based on previously seen training cases. If additional causal or diagnostic knowledge is available to the agent, this search can be further focused. In our example, plans that include the push-button operator are preferred over plans that do not include operators associated with turning on lights. If the agent has no previous case knowledge and no other guidance, this search defaults to iterative deepening.

Identifying the cause of the error and learning a correction

There are two credit assignment problems to be solved in identifying and correcting an error in the agent's knowledge: first, identifying which operator is incorrect and second, identifying how that operator's knowledge is incorrect.

IMPROV identifies which operator(s) have incorrect knowledge by comparing the successful plan to the original (incorrect) plan. Ordering and instantiation differences between the plans indicate potentially incorrect operators. In our simple light example, both the correct and incorrect plans contain a single operator (push-button) with different instantiations, so it is identified as the erroneous operator.

Once the operators with incorrect knowledge have been identified, IMPROV uses an extension of the incremental category learner, SCA (Miller 1991; 1993), to identify how the operator is incorrect. The category being learned is which operator to select for a given state and goal. To avoid the error in future, the agent must decide which features of the state or goal caused the operator to succeed or fail. In our example, the agent must learn that the reason pushing the red button works is because the button is red, not because it is the button on the right, or because the robot's hand is open or closed etc.[2] In its weakest form, IMPROV simply relies on the induction made by SCA to determine the features that are responsible for the failure. It revises the reflective knowledge accordingly and replans to correct the original error in the agent's deliberate knowledge. The inductive guess may be wrong, leading to a future error and the need for another recovery.

Instruction to inform recovery

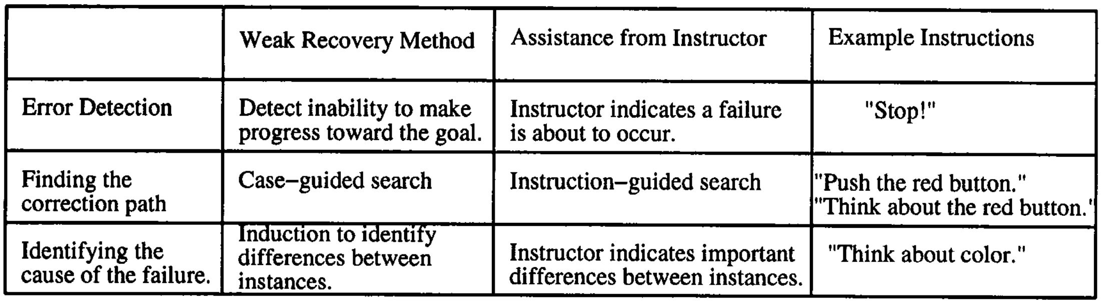
We have augmented IMPROV by allowing an instructor to interrupt the recovery process at any time and provide instructions that guide the recovery. Figure 4 lists how instructions can apply to each phase of recovery. In particular, instructions can be used to warn the agent that an error is about to happen, to help the agent find the correct way to achieve its current goal or to help identify the reasons for the error.

Error detection

IMPROV's instructor can indicate that an error is about to happen by interrupting deliberate processing with the command "Stop!". Our agent assumes that this command means that the currently selected operator will lead to a execution error, rather than to the current goal. In our example, if the agent chooses the wrong (green) button, the instructor may say "Stop", realizing an error is about to occur as the agent moves it's hand over the wrong

[1] We currently assume that a successful plan exists and therefore the agent does not have to determine when to abandon an impossible task.

[2] This explanation is limited by the agent's representation language. To determine the real cause, that the red button is electrically connected to the light, the agent would need a deeper theory.



	Weak Recovery Method	Assistance from Instructor	Example Instructions
Error Detection	Detect inability to make progress toward the goal.	Instructor indicates a failure is about to occur.	"Stop!"
Finding the correction path	Case–guided search	Instruction–guided search	"Push the red button." "Think about the red button."
Identifying the cause of the failure.	Induction to identify differences between instances.	Instructor indicates important differences between instances.	"Think about color."

Figure 4: The stages of a recovery and how instruction can help.

button. The system then treats this situation as if an error had occurred and begins the recovery process. IMPROV records the operator push-button(green), along with the state when the operator was chosen, as a negative training instance for the inductive module and starts a search for the correct way to turn on the light.

Finding a plan that reaches the goal

Once an error has been detected, IMPROV searches for a correct plan to achieve the current goal. At any time during this search, the instructor may provide guidance in the form of one or more steps in that plan. In our example, the instructor can interrupt the search by saying:

> "Push the red button."

This leads the agent to prefer plans that include pushing the red button. As the agent believes pushing any button turns on the light, it believes this plan will succeed and immediately executes it. After it succeeds, the agent's knowledge is corrected just as if it had discovered the plan through search instead of instruction. In the event that the instructions are incorrect, IMPROV will try the suggested path, see that it fails and continue searching for a correct plan.

Identifying the cause of the error and learning a correction

Once a correct plan has been found (either through search or with the aid of instruction), IMPROV needs to identify the cause of the error. IMPROV must determine the key difference(s) between the plan that succeeded and the plan that failed, which in general is a difficult credit assignment problem. In our example, the green button is to the right of the red button. Without help from the instructor, IMPROV must guess whether the reason push-button(green) failed was because the button was on the right, or because it was green. If it makes the wrong induction, the agent may fail later. However, the instructor can interrupt the learning process and guide the choice of features for the induction. In our example, if the instructor says:

> "Think about color."

this leads IMPROV to focus on the colors of the buttons (rather than their positions) and ensures that the correct induction is made.

		Destination Level		
		Reflex Level	Deliberate Level	Reflective Level
Source Level	Reflex Level	RL/Empirical (Refinement)	?? (Synthesis)	?? (Synthesis)
	Deliberate Level	?? (Compilation)	Instructo-Soar/Analytic IMPROV/Empirical (Refinement)	?? (Synthesis)
	Reflective Level	EBNN / Analytic & Empiric EBRL / Analytic & Empiric EBC / Analytic & Empiric (Compilation)	Instructo-Soar/Analytic IMPROV/Empirical (Compilation)	Instructo-Soar/Analytic IMPROV/Empirical EITHER/Empirical EXPO/Empirical (Refinement)

Figure 5: Matrix of Learning between Levels

Conclusion

The thrust of this paper has been to explore adaptation in multi-level agents. Our current work has made some important inroads by demonstrating the feasibility of using analytic and empirical processing at the reflective and deliberate level to improve the agents abilities at those levels. The agents are able to learn new things about their environments that would not be possible in a single level agent. However, there is still more to be done before we have agents that can adapt at all levels from all available sources of knowledge.

Figure 5 is a matrix that attempts to classify current progress in learning across levels using both analytic and empirical techniques. Each row contains the problems of transferring knowledge from one specific level to another level. Implicit in these rows is that the knowledge in that level is combined with knowledge from the external environment, thus learning from the reflex level does not just use the agent's current reflex knowledge at that level, but uses that knowledge together with its experiences in its environment to improve that level. The columns are the destination of the knowledge — where the learning happens. Many systems appear in multiple columns because learning at one level immediately transfers to the higher levels. That is true in both IMPROV and Instructo-Soar. One weakness in this figure is that it does not identify specific learning problems that cross all levels. For example, a particularly challenging area is the learning of new representations for tasks.

There are still some missing entries, and even when there is an entry, there are undoubtably many research issues. Furthermore, additional research is needed on how to create agents that integrate methods from the complete matrix. Our hope is that the workshop will fill in some of the missing entries (that we are not aware of), but also stimulate the community to look at integration of multiple techniques to help create more complete multi-level adaptive intelligent agents.

References

DeJong, G. 1995. A case study of explanation-based control. In *Proceedings of the Twelth International Workshop on Machine Learning*, 167–175.

Dietterich, T. G., and Flann, N. S. 1995. Explanation based learning and reinforcement learning: A unified view. In *Proceedings of the Twelth International Workshop on Machine Learning*, 176–184.

Gil, Y. 1991. A domain-independent framework for effective experimentation in planning. In *Proceedings of the International Machine Learning Workshop*, 13–17.

Huffman, S. B., and Laird, J. E. 1994. Learning from highly flexible tutorial instruction. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*.

Huffman, S. B. 1994. *Instructable autonomous agents.* Ph.D. Dissertation, University of Michigan, Dept. of Electrical Engineering and Computer Science.

Laird, J. E. 1988. Recovery from incorrect knowledge in Soar. In *Proceedings of the National Conference on Artificial Intelligence*, 618–623.

Miller, C. M. 1991. A constraint-motivated model of concept formation. In *The Thirteenth Annual Conference of the Cognitive Science Society*, 827–831.

Miller, C. M. 1993. *A model of concept acquisition in the context of a unified theory of cognition.* Ph.D. Dissertation, The University of Michigan, Dept. of Computer Science and Electrical Engineering.

Mitchell, T. M., and Thrun, S. B. 1993. Explanation based learning: A comparison of symbolic and neural network approaches. In *Proceedings of the Tenth International Workshop on Machine Learning*, 197–204.

Mitchell, T. M.; Keller, R. M.; and Kedar-Cabelli, S. T. 1986. Explanation-based generalization: A unifying view. *Machine Learning* 1.

Ourston, D., and Mooney, R. J. 1990. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the National Conference on Artificial Intelligence*, 815–820.

Pearson, D. J., and Laird, J. E. 1996. Toward incremental knowledge correction for agents in complex environments. In Muggleton, S.; Michie, D.; and Furukawa, K., eds., *Machine Intelligence*, volume 15. Oxford University Press.