Learning to troubleshoot: Multistrategy learning of diagnostic knowledge for a real-world problem solving task[*]

Ashwin Ram, College of Computing, Georgia Tech

S. Narayanan, School of Industrial and Systems Engineering, Georgia Tech

Michael T. Cox, College of Computing, Georgia Tech

Abstract

This article presents a computational model of the learning of diagnostic knowledge based on observations of human operators engaged in a real-world troubleshooting task. We present a model of problem solving and learning in which the reasoner introspects about its own performance on the problem solving task, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies. The model is implemented in a computer system which provides a case study based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model is intended as a computational model of human learning; in addition, it is computationally justified as a uniform, extensible framework for multistrategy learning.

[*]Technical Report GIT-CC-93/67, College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280, 1993. E-mail: `ashwin.ram@cc.gatech.edu` .

Contents

1 Introduction

This article presents a case study of multistrategy learning for the problem of learning diagnostic knowledge during a troubleshooting task. The case study is based on observations of human operators engaged in a real-world troubleshooting task. We present a computational model of problem solving and learning in which the reasoning system introspects about its own performance on the problem solving task, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies.

This research was motivated by two considerations. First, although there has been a significant growth of research on machine learning, much of this research has not been performed in the context of complex real-world problem solving tasks (cf. [Riddle, 1992]). As a result, the issues of scalability and robustness of these methods, as they are applied to real-world problems, are still unresolved in many cases. To promote the applicability and usability of research methods, it is important to ground theories of reasoning, knowledge representation, and learning in the context of real-world tasks and domains.

Our second motivation was to provide a computational account of human learning in the context of a real-world problem. The model presented in this article is based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model is implemented in a computer system, Meta-TS,[1] which uses multiple types of knowledge to troubleshoot printed-circuit boards that fail in the test area of the manufacturing plant. Meta-TS has been evaluated on a series of troubleshooting problems, including actual problems encountered by the human operators in the manufacturing plant. The underlying model is intended as a computational model of human learning; in addition, it is computationally justified as a uniform, extensible framework for multistrategy learning in machine learning systems.

1.1 The problem

One of the critical areas in electronics assembly manufacturing is the test and repair area [Douglas, 1988; Kakani, 1987]. It is estimated that about 20% of manufactured printed-circuit boards (PCBs)

[1]TS stands for TroubleShooter.

fail in the test area in a typical electronics assembly line. When PCBs spend a considerable amount of time in the test and repair area, it increases the work-in-process inventory and slows down the feedback to the manufacturing line necessary for achieving better process control. This results in significant deterioration of system performance. Computerized decision aids can potentially alleviate some of the major problems in the test and repair area and facilitate enhanced system performance. A key to developing computer-based aids is understanding the human problem solving processes that carry out the complex task of troubleshooting in an assembly line situation. While there has been much interest in developing AI applications in various areas of electronics manufacturing (see, for example, [Miller and Walker, 1988]), most of this research has not dealt with the issues of learning or cognitive modelling.

NCR's manufacturing plant located near Atlanta has state-of-the-art facilities in electronics assembly manufacturing with a newly installed surface mount technology (SMT) line. Our project began when the plant became operational in January 1990 [Cohen, 1990]. At that time, the plant was facing typical start-up problems experienced by most new facilities. There were a high number of PCBs in the test and repair region waiting to undergo troubleshooting, resulting in high work-in-process inventories. Our analysis of the system revealed that developing a model of the troubleshooting operator would provide a structure for designing and implementing computer-based tools for this task. The effort would also facilitate formalization of the task, which could then be used in the design of instructional systems [Clancey, 1986], thereby facilitating the development of a flexible work force that is complementary to the policy of the company.

A schematic of the manufacturing plant is shown in figure 1. An unpopulated board enters the SMT line where it is populated with components, soldered, cleaned, and sheared. The populated board then enters the test and repair area. A typical PCB manufacturing line has two major test and repair areas, the "in-circuit test" area and the "functional test" area. Boards are shipped only after they pass both the in-circuit test (ICT) area and the functional test area. Our research focusses on the troubleshooting process when a board fails in the ICT area.

In the ICT area, the populated board is mounted on an automated ICT machine. The ICT machine checks individual components as well as connections between components for proper functioning through several test procedures. If the board passes the tests, an appropriate message appears on the console of the ICT machine. If the board fails any of the tests, the ICT machine produces a ticket listing the detected failure(s). For example, a component on the board could fail to meet the desired specifications, known as the "nominal" values of the component's parameters.

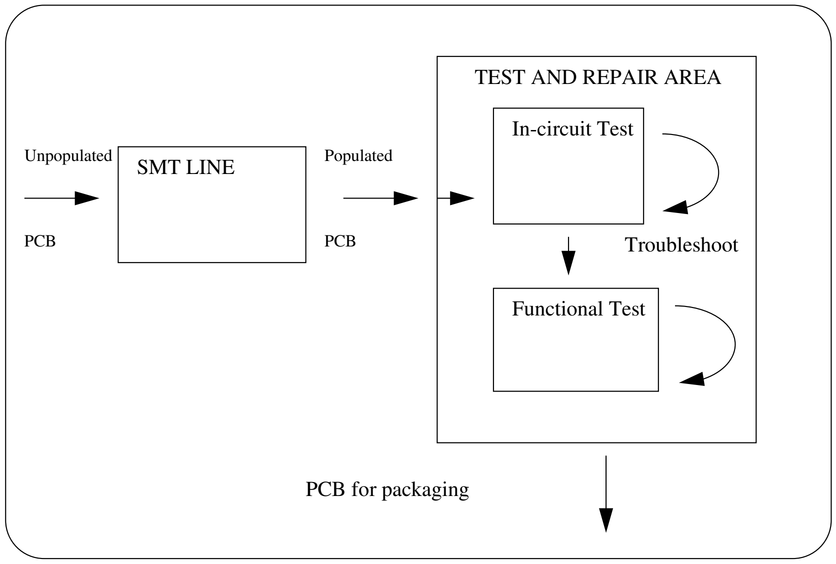

Figure 1: A schematic of the NCR electronics manufacturing plant.

The ICT machine may also provides additional symptomatic information which can be used by the human operator in the troubleshooting process. The operator then uses the information in the ticket to troubleshoot the board. Troubleshooting is a complex task which can be broken into two components: diagnosis and repair. Diagnosis is a problem solving task in which the operator arrives at a description of cause of the failure and identifies an appropriate set of repair actions to fix the faulty PCB. Repair involves carrying out the repair actions (in this case, usually a set of manual actions performed by the operator on the board).

We developed a computational model of an operator involved in the task of troubleshooting a faulty PCB. The model was based on protocol analysis of over 300 problem solving episodes gathered in the ICT area of the NCR plant [Cohen, Mitchell, and Govindaraj, 1992], and implemented in a computer system that performed the troubleshooting task. Of the data collected, one set was used in the development of the model and the remaining set was used to perform both behavioral validation and process validation of the computational model. We found that although the problem solving model was a fair representation of a skilled troubleshooting operator, it had some limitations. First, the system assumed that its knowledge was correct and complete during the reasoning process. It is difficult to hand code all the knowledge required for this task. Furthermore, even if this could be done, the system would still be faced with the "brittleness" problem. Due to the dynamics of the system state changes in the electronics manufacturing domain, the computational model must be flexible and robust. For example, one of the pieces of knowledge in the system was "Resistor r254 is often damaged." This occurred due to a process problem in the manufacturing plant. If the process problem were fixed, the association would no longer be valid. The system must have the capability of altering its world model to reflect changes in the real world. In addition, the problem solving model did not capture improvement in the problem solving skills of the troubleshooting operator. Thus, the model was incomplete as a cognitive model of human troubleshooting.

These considerations motivated our research towards incorporation of a learning model in the system. The complete problem solving and learning system is fully implemented in the Meta-TS program, which has been evaluated using the data gathered at the NCR plant. In this article, we will focus primarily on the learning aspects of the system. The underlying model is intended both as a cognitive model of human learning during the troubleshooting task, and as a computational solution to the brittleness problem which is faced by all systems in a dynamic world.

1.2 The approach

It is generally accepted that learning is central to intelligent reasoning systems that perform realistic reasoning tasks, such as understanding natural language stories or solving non-trivial problems (e.g., [Schank, 1983]). It is impossible to anticipate all possible situations in advance and to hand-program a machine with exactly the right knowledge to deal with all the situations that it might be faced with. Rather, during the performance of any non-trivial reasoning task, whether by human or by machine, there will always be failures. An important aspect of intelligence lies in the ability to recover from such failures and, more importantly, to learn from them so as not to make the same mistake in future situations.

Machine learning research over the past few years has focussed on the development of learning algorithms for particular classes of problems. These algorithms are tailored to particular problems including inductive learning (e.g., induction of decision trees [Quinlan, 1986], conceptual clustering [Michalski and Stepp, 1983; Fisher, 1987]), analytical learning (e.g., explanation-based learning [DeJong and Mooney, 1986; Mitchell, Keller, and Kedar-Cabelli, 1986], learning from explanation failures [Hall, 1988; VanLehn, Jones, and Chi, 1992]), and analogical learning (e.g., analogy [Falkenhainer, 1989; Gentner, 1989], case-based learning [Carbonell, 1986; Hammond, 1989]). Recently, there has been much interest in "multistrategy learning," and several approaches have been suggested (see, e.g., [Michalski and Tecuci, in press]). These approaches fall into four broad categories, which we call strategy selection models, toolbox models, cascade models, and single mechanism models.

The common element in all these approaches is the use of multiple learning methods to allow the reasoning system to learn in multiple types of learning situations. In *strategy selection models*, the reasoner has access to several learning strategies, each represented as a separate algorithm or method. Learning involves an explicit decision stage in which the appropriate learning strategy is identified, followed by a strategy application stage in which the corresponding algorithm is executed. Methods for strategy selection also differ. Pazzani's [1991] OCCAM system, for example, tries each learning strategy in a pre-defined order until an applicable one is found; Reich's [in press] BRIDGER system uses a task analysis of the problem solving task to determine the appropriate learning strategies for each stage of the task; Hunter's [1990a] INVESTIGATOR system represents prerequisites for application of each learning strategy; and Ram and Cox's [in press] Meta-AQUA system uses characterizations of reasoning failures to determine what to learn and, in turn, the learning strategies to use to learn it.

Toolbox models are similar to strategy selection models in that they too incorporate several learning strategies in a single system. The difference is that these strategies are viewed as tools that can be invoked by the user to perform different types of learning. The tools themselves are available for use by other tools; thus, learning strategies may be organized as coroutines. An example of this approach is Morik's [1991] MOBAL system, in which learning occurs through the cooperation of several learning tools with each other and with the user. In *cascade models*, two or more learning strategies are cascaded sequentially, with the output of one strategy serving as the input to another. For example, Danyluk's [1991] GEMINI system uses a cascade of explanation-based learning, conceptual clustering, and rule induction strategies, in that order, to combine analytical and empirical learning into a single learning system. Clearly, these categories of models are not exclusive of each other (e.g., a strategy selection system may choose to cascade learning strategies in certain circumstances), but they serve to characterize the major ways in which learning strategies may be integrated. Finally, *single mechanism models* use a single underlying mechanism as a "weak method" which can perform different types of learning depending on the situation. Examples of such models are Laird, Rosenbloom and Newell's [1986] SOAR, and Tecuci and Michalski's [1991] MTL. Even in such systems, it is still important to characterize the learning strategies that are implemented by (or that emerge from) the single mechanism, and the circumstances under which different strategies are used by the system.

The Meta-TS system presented in this article is based on the strategy selection model of the Meta-AQUA system. Our approach uses a set of available learning strategies that are selected through an introspective analysis of the system's reasoning processes. This introspection results in the formulation of explicit learning goals which may then be achieved by invoking an appropriate learning strategy or even deferred until a later learning opportunity arises. In particular, we propose a meta-explanation structure to represent classes of learning situations along with the types of learning needed in those situations. This structure aids in the analysis of a trace of the system's reasoning processes to identify the nature of the reasoning failure. Our method, called *introspective multistrategy learning*, combines meta-reasoning with multistrategy learning to allow the system to determine what it needs to learn and how that learning should be performed.

The focus of our research is on the integration of different kinds of knowledge and reasoning processes into real-world systems that can learn through experience. In particular, we are interested in modelling active, goal-driven learning processes that underlie deliberative learning during the performance of complex reasoning tasks. The introspective process in our model relies on representations of meta-explanations about reasoning. These are similar to self-explanations [Chi and VanLehn, 1991; Pirolli and Bielaczyc, 1989; Pirolli and Recker, in press; VanLehn, Jones, and Chi, 1992], with the difference that self-explanations are explanations about events and objects in the external world, whereas our meta-explanations are explanations about events and objects in the reasoner's train of thoughts—the mental world. While experimental results in the metacognition literature suggest that introspective reasoning of the kind we propose here can facilitate reasoning and learning (see, for example, [Schneider, 1985; Weinert, 1987]), it is important to develop computational models that specify the mechanisms by which this facilitation occurs and the kinds of knowledge that these mechanisms rely on. In this article, we will present such a computational model of human learning based on our theory of introspective multistrategy learning.

Our approach is motivated by computational and system design considerations as well. The approach relies on a declarative representation of meta-models for reasoning and learning. There are several advantages of maintaining such structures in memory. Because these structures represent reasoning processes explicitly, the system can directly inspect the reasons underlying a given processing decision it has taken and evaluate the progress towards a goal. Thus, these representations can also be used in credit and blame assignment, to analyze why reasoning errors occurred, and to facilitate learning from these errors. Furthermore, these knowledge structures provide a principled basis for integrating multiple reasoning and learning strategies, and the unified framework makes it relatively straightforward to incorporate additional types of learning situations and additional learning strategies for these situations.

We now discuss the technical details of the computational model, including the problem solving system and the introspective multistrategy learning system that constitute the major components of the model.

2 The diagnostic problem solving module

A schematic of the problem solving system, the troubleshooting module of Meta-TS, is shown in figure 2 [Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992]. The module takes as input the ICT ticket information and the PCB information. The output of the module is a diagnosis and a set of recommended repair actions. The problem solving process uses various types of knowledge, troubleshooting actions, and control methods, briefly discussed below.

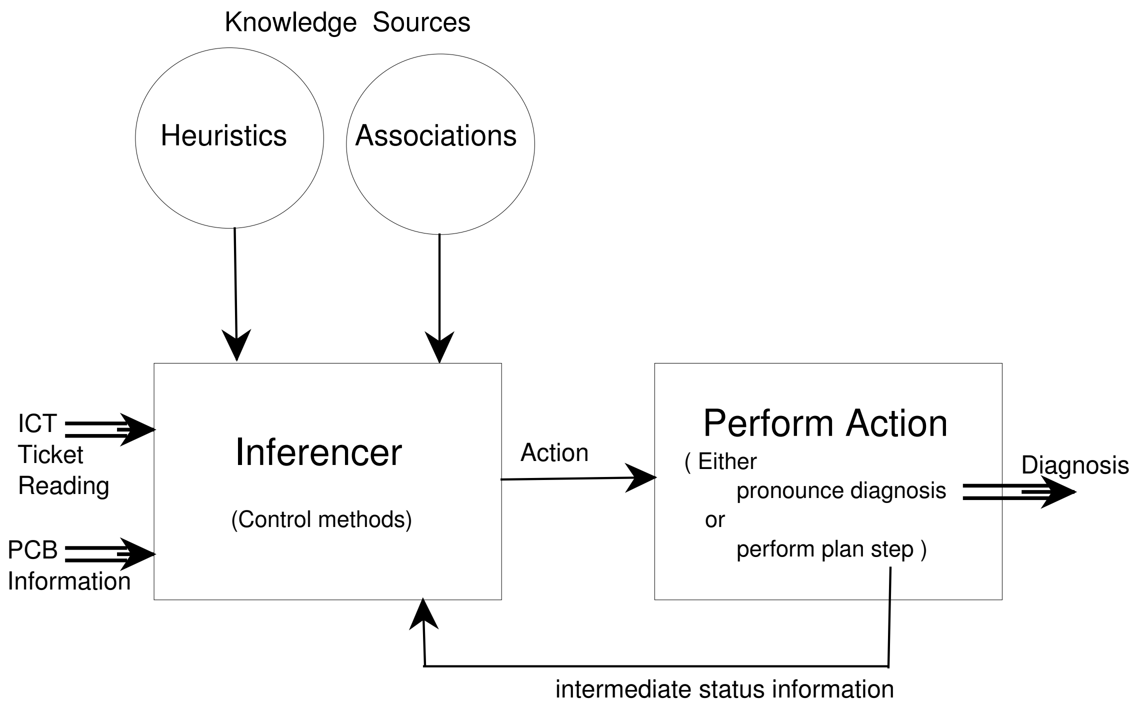

Figure 2: The problem solving module for the troubleshooting task.

The problem solving module uses different types of knowledge as well as available real-world troubleshooting actions to hypothesize the cause of a failure and to suggest repair actions for that failure. Based on our data from the human operators in the NCR plant, we categorized diagnostic knowledge into two broad types, associations and heuristics [Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992]. *Associations* are simple rules which directly map a particular symptom to a specific diagnosis. The operator may perform an intermediate action to confirm the hypothesis, but usually does not perform a series of search sequences. This type of knowledge is context-sensitive and is indexed by board type. *Heuristics* are standard rules of thumb. These rules are not context-sensitive and are applicable across board types. Heuristics are used by the operator for troubleshooting when there is no known association for a given problem situation. This knowledge determines the series of standard operating procedures performed in troubleshooting a faulty PCB. Some examples of associative and heuristic knowledge in the system are shown in table 1.

In addition to associative and heuristic knowledge, the problem solving module can also use *troubleshooting actions*, which are intermediate subtasks performed by the system to gather the board information. These correspond to explicit operator actions used in gathering information and confirming intermediate hypotheses. Finally, the *control methods* in the problem solving module are procedures that enable the system to look at the symptoms, utilize the appropriate type of knowledge, invoke proper intermediate actions, and finally arrive at the diagnosis result. This result, also called a "diagnostic," is a description of the failure along with the *repair action(s)* necessary to fix the faulty PCB. It is also possible that the ICT reading is incorrect (known as a "bogus" reading), in which case the PCB is rerun through the ICT machine. Some examples of troubleshooting actions, control methods, and repair actions are shown in table 2.

The problem solving module was initially implemented as a separate system, and then later incorporated into Meta-TS along with the multistrategy learning module. The implementation used AT&T 2.1 C++ on a SUN workstation under the UNIX operating system. The ICT ticket information and the PCB information were represented as C++ classes. Class representations were also used for associative and heuristic knowledge, control methods, troubleshooting actions, and repair actions in the system. An example of a problem solving episode is shown in figure 3.

In order to validate the troubleshooting process of the problem solving module, we added an explanation facility to keep track of the system's problem solving process and produce a trace of the problem solving at the end of each problem solving episode. The problem solving traces were compared with the verbal protocol data gathered from the human operators at the assembly

Associative knowledge
• r254 is often damaged. Visually inspect the part. If it is damaged, replace the part.
• If r1 or r2 fails, the ticket reading is bogus. Output the diagnosis "Bogus ICT ticket."
• u56 and u65 are known bad parts. Use the "lifted leg" procedure to identify the bad part(s) and replace them.
• r228, r239 and r279 are connected to u51. If one of these has failed with a low reading, u51 should be replaced.
Heuristic knowledge
• If the measured reading of a resistor is slightly higher than the nominal value on the ICT ticket, perform the "visual inspection" procedure. If the defect is found, terminate the search, otherwise output the diagnosis "Unable to make an inference" and perform appropriate repair action.
• If the measured reading of the resistor is slightly lower (qualitatively) than the nominal value, perform the "ohming out" action. If the diagnosis is "Bogus ICT ticket", terminate the search, otherwise perform the "check schematics" action and make an ordered list of faulty ICs. If any of these can be fixed by association-based search, terminate search, otherwise test each of these ICs to determine the faulty component. If a defective component is not found, terminate the search and output the diagnosis "Unable to make an inference."

Table 1: Examples of associative and heuristic knowledge used in the problem solving module. r# indicates the number of a resistor component, and u# indicates the number of an IC (integrated circuit) component.

```
Troubleshooting board #6

ICT ticket information
    ------------------------
    These are board faults
    TA 7052 MAX Processor
    ------------------------
    r243 has failed
    Measured  = 18.000000  ohms
    Nominal   = 10.000000  ohms
    ------------------------

Entering problem solver

Getting symptom information from ticket
  r243 has failed

Looking for associations for r243
  No associations found

Association search unsuccessful
Diagnosing by heuristics

Looking for heuristics

Applying heuristic-3
  Measured value is much higher than nominal value
  Suspecting an open/defective part

Ohming out on r243
  Ticket reading verified

Performing visual inspection
  Defective part verified

Diagnosis: Defective part, r243 is defective
Repair action: Replace r243
```

Figure 3: An example of a problem solving episode.

Troubleshooting actions

- Perform visual inspection.
- Check for faulty IC that lowers resistance.
- Ohm out on a resistor.
- Check schematics.

Control methods

- Look at the symptom information on the ICT ticket first.
- Determine if there is an association for that symptom in memory; if so, invoke it and terminate the search.
- Perform the "visual inspection" action. If the defect is found, suggest appropriate repair action and terminate search. Use the appropriate heuristics and determine the repair action depending on the qualitative difference between the measured and nominal reading in the ticket.

Repair actions

- Identify the part number of the defective part and replace it with an equivalent part.
- Output "Bogus ICT ticket reading" to indicate a suspected false ICT ticket reading.
- Identify the part number of the missing part and install an appropriate part.
- Output "Unable to make an inference" to indicate insufficient knowledge to arrive at an inference that indicates a repair action.

Table 2: Examples of troubleshooting actions, control methods, and repair actions in the problem solving module.

plant. (These problem solving traces also played a central role in the learning module; this will be discussed in more detail in section 3.) Using the problem solving traces, the problem solving module was validated on 75% of the problem solving episodes in our data for a major category of board failures. On 84% of these episodes, the model arrived at the same diagnostic result as an operator did in the real world for the same input information; and on 68% of the episodes, similar actions were performed in the solution process [Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992]. This article focusses on the learning module; further details of the problem solving module can be found in [Cohen, 1990; Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992].

3 The introspective multistrategy learning module

Although the results from the standalone model of troubleshooting showed that the problem solving module was a reasonably good model of a skilled troubleshooting operator, there were also some obvious shortcomings. Electronics manufacturing, like most other real-world domains, is a complex and highly dynamic process. A complete model of troubleshooting in such a domain requires a large amount of knowledge; in addition, the operator's knowledge will be inherently incomplete and subject to change as the process being modelled changes. The problem solving module, in contrast, was based on the assumption that the available knowledge was complete and correct; it did not have the flexibility necessary to deal with this task domain over an extended period of time. Furthermore, the model failed to capture the improvement of problem solving skills through experience, an important aspect of human performance in any task domain.

For these reasons, we developed a learning module that allowed the system to learn incrementally from each problem solving episode. This module was based on observations of troubleshooting operators in the plant, protocol analysis of the problem solving process, and critical examination of the computational model of the troubleshooting operator as implemented in the problem solving module. The overall system, called Meta-TS, uses multiple learning strategies, both supervised and unsupervised, and a strategy selection mechanism to invoke the appropriate strategies in different situations. Supervised learning occurs in situations in which a novice troubleshooter gets explicit input from a skilled troubleshooter (the supervisor). In unsupervised learning, a troubleshooter adapts his or her domain knowledge based on problem solving experience without expert input.

Since a particular problem solving episode may involve several pieces of knowledge (potentially of different types), the troubleshooter, whether human or machine, must be able to examine the reasons for successes and failures during problem solving in order to determine what needs to be learned. For example, if the system fails to arrive at a correct diagnosis, it needs to determine which piece of knowledge was missing or incorrect. To accomplish this, the system must be able to examine its own problem solving processes. Thus, the problem solving traces discussed earlier are a crucial component of the computational model of multistrategy learning.

Meta-TS uses declarative representations of the knowledge and methods used for problem solving in order to facilitate critical self-examination. A trace of the problem solving process is constructed during the troubleshooting episode, and introspectively analyzed during the learning phase to determine what the system might learn from that episode. The analysis also helps the system select the learning strategy appropriate for that type of learning.

3.1 What is to be learned?

Since the problem solving module relies on associative and heuristic knowledge, the learning module must, in general, be able to acquire, modify, or delete such associations and heuristics through experience. In order to be more specific about the constraints on and output of the learning task, it is necessary to examine the troubleshooting model in more detail. Recent research in diagnostic problem solving has proposed the use of "deep" reasoning methods [Richardson, Keller, Maxion, and Polson, 1985] or integration of "deep" and "shallow" reasoning methods in knowledge-based systems [Fink and Lusth, 1987] and in tutoring systems [Lesgold, Lajoie, Bunzo, and Eggan, 1988]. Our observations revealed that operators rely predominantly on "shallow" reasoning methods using heuristic and context-sensitive associative knowledge during problem solving [Cohen, 1990; Cohen, Mitchell, and Govindaraj, 1992; Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992]. This may be due to the fact that the ICT machine filters out most of the topographic knowledge of the PCB and causal knowledge of the components in the board through a series of tests. The observation by Barr and Feigenbaum [1981], that humans often solve a problem by finding a way to think about the problem that facilitates the search for a solution, was clearly evident in our study. In this task domain, the search is carried out through "shallow" reasoning using associations and heuristics; furthermore, the search is sensitive to process changes and can sometimes make use of a human expert. Thus, the learning strategies implemented in Meta-TS focus on the supervised and unsupervised acquisition, modification, and deletion of associative knowledge through the analysis of reasoning traces which, however, do not contain detailed domain knowledge.

Associative knowledge improves the system performance in two ways. First, it improves the speed of the problem solving process. Using associative knowledge typically results in the reduction of some intermediate steps in the reasoning process, thus resulting in some savings in the time required to troubleshoot. This is important for assembly line tasks which are typically highly time-constrained. Second, associative knowledge can provide solutions in cases where heuristic knowledge requires information about the board that is not easy to obtain. In general, associative knowledge contributes to the quality and correctness of the solution for a large number of the problem solving situations. This was evident in our data from the electronics assembly plant, and has also been observed by other researchers (e.g., [Arabian, 1989]). Thus, an important type of learning is one in which the operator learns associations through experience.

Human operators involved in troubleshooting also appear to learn some heuristic knowledge. We noticed that the training program for novice human operators primarily focusses on manual skills such as soldering and performing actions such as ohming out. However, the problem solving process of skilled human operators in the plant revealed that they often use certain standard operating procedures or heuristics. The source of this heuristic knowledge appears to be the result of generalization of associations learned over time while troubleshooting. In addition, as is the case for associations, heuristics can be learned through both supervised and unsupervised learning methods. The current implementation of Meta-TS focusses on the learning of associative knowledge through experience and does not include strategies for learning heuristics. More research is needed to develop such strategies; however, once developed, it would be relatively easy to incorporate them into the system due to the modular design of the multistrategy learning system.

3.2 Architecture of the learning module

Our approach to multistrategy learning is based on the analysis of declarative traces of reasoning processes to determine what and how to learn [Ram and Cox, in press]. A particular troubleshooting episode may involve many different associations, heuristics, and troubleshooting actions. If the final diagnosis is incorrect, the system must analyze its reasoning process, assign blame for its failure, and determine what it needs to learn in order to avoid making a similar mistake again in the future. If the diagnosis is correct, the system can determine what it might learn in order to improve the process that led up to this diagnosis. Finally, depending on the type of learning that is necessary, the system must invoke an appropriate learning strategy. Thus, learning is viewed as a deliberative, planful process in which the system makes explicit decisions about what to learn and how to learn it [Hunter, 1990b; Quilici, to appear; Ram, 1991; Ram and Hunter, 1992; Ram and Leake, in press; Redmond, 1992]. In our introspective multistrategy learning framework, these decisions are based through introspective analysis of the system's performance, which relies on meta-knowledge about the reasoning performed by the system during the performance task, about the system's knowledge, and about the organization of this knowledge [Ram and Cox, in press; Ram, Cox, and Narayanan , in press].

The key representational entity in our learning theory is a *meta-explanation pattern* (Meta-XP), which is a causal, introspective explanation structure that explains how and why an agent reasons, and which helps the system in the learning task [Cox and Ram, 1991; Cox and Ram, 1992b; Ram and Cox, in press]. There are two broad classes of Meta-XPs. *Trace Meta-XPs* record a declarative trace of the reasoning performed by a system, along with causal links that explain the decisions taken. The trace holds explicit information concerning the manner in which knowledge gaps are identified, the reasons why particular hypotheses are generated, the strategies chosen for verifying candidate hypotheses, and the basis for choosing particular reasoning methods for each of these. Trace Meta-XPs are similar to "reasoning traces" [Carbonell, 1986; Minton, 1988; Veloso and Carbonell, 1991] or "justification structures" [Birnbaum, Collins, Freed, and Krulwich, 1990; deKleer, Doyle, Steele, and Sussman, 1977; Doyle, 1979], with the difference that Trace Meta-XPs represent, in addition to the subgoal structure of the problem and justifications for operator selection decisions, information about the structure of the (possibly multistrategy) reasoning process that generated a solution. For example, at the highest level of granularity, a node in a Trace Meta-XP might represent the choice of a reasoning method such as association-based search or heuristic reasoning, and at a more detailed level a node might represent the process of selecting and using a particular association or heuristic. These structures could, therefore, be viewed as representing the "mental operators" underlying the reasoning process.

The major contribution of our approach, however, is the use of a new kind of meta-explanation structure to represent classes of learning situations along with the types of learning needed in those situations. This structure, called an *Introspective Meta-XP*, aids in the analysis of the reasoning trace to analyze the system's reasoning process, and is an essential component of a multistrategy learning system that can automatically identify and correct its own shortcomings. Thus, instead of simply representing a trace of the reasoning process, we also represent the knowledge required to analyze these traces in order to determine what to learn and how to learn it. After a problem solving episode, the system uses Introspective Meta-XPs to examine the declarative reasoning chain in order to both explain the reasoning process and to learn from it. These structures associate a failure type[2] with learning goals and the appropriate set of learning strategies for pursuing those goals. Thus, an Introspective Meta-XP performs three functions: it aids in blame assignment (determining which knowledge structures are missing, incorrect or inappropriately applied); it aids in the formulation of appropriate learning goals to pursue; and it aids in the selection of appropriate learning algorithms to recover and learn from the reasoning error. Such meta-explanations augment a system's ability to introspectively reason about its own knowledge, about gaps within this knowledge, and about the reasoning processes which attempt to fill these gaps. The use of explicit Meta-XP structures allows direct inspection of the reasons by which learning goals are posted and processed, thus enabling a system to improve its ability to reason and learn.

The modules and the control flow in the introspective multistrategy learning module are shown in figure 4. The problem solving module has a declarative representation of the associative knowledge used in troubleshooting. The learning module can add, delete, or modify associative knowledge in the problem solving module. During a troubleshooting episode, a trace of the reasoning performed by the system along with causal links that explain the intermediate decisions taken is recorded in an instance of a Trace Meta-XP. The introspector takes the reasoning trace as input. It also has methods to perform tests on the world, allowing the system to perform experiments to help in hypothesis generation and verification [Rajamoney, 1989]. These methods access the ICT information and the PCB information which is input to the problem solving system. The introspector has a set of verification actions that includes methods to obtain statistical information and to gather information from an expert troubleshooter. There is also a set of learning strategies represented declaratively along with information for strategy selection. In Meta-TS, the introspector is implemented as a C++ class with methods for each learning strategy (see figure 5); this class encodes the knowledge that corresponds to the Introspective Meta-XPs discussed earlier. The learning strategies currently implemented in Meta-TS are discussed in the next section.

After every problem solving episode, the introspector looks at the reasoning trace and uses information gathered from the tests on the world to determine if the system can learn something from this experience. Learning occurs when the system fails to make the correct diagnosis (due to

[2]Note that a correct solution that is inefficiently produced is still a reasoning "failure."

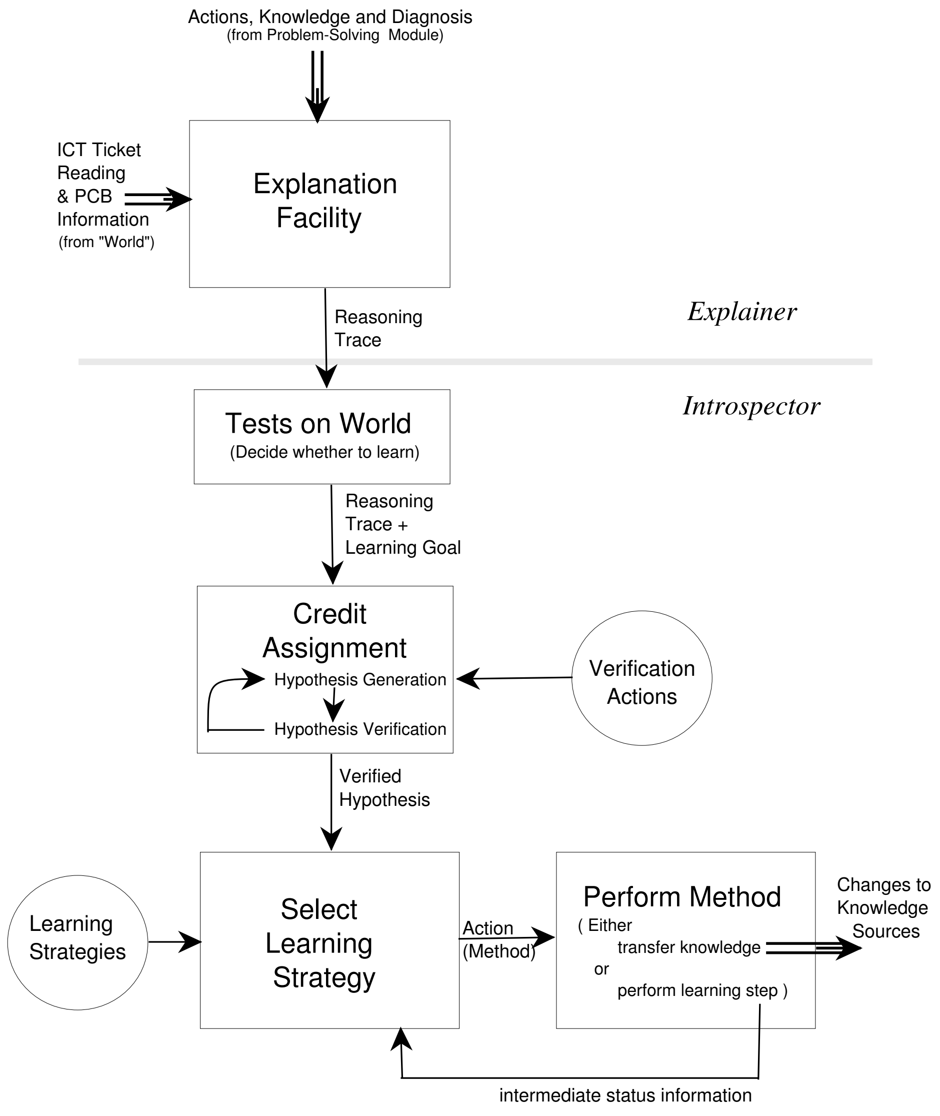
Actions, Knowledge and Diagnosis
(from Problem-Solving Module)
ICT Ticket Reading & PCB Information
(from "World")
Explanation Facility
Reasoning Trace
Explainer
Introspector
Tests on World
(Decide whether to learn)
Reasoning Trace + Learning Goal
Credit Assignment
Hypothesis Generation
Hypothesis Verification
Verification Actions
Verified Hypothesis
Learning Strategies
Select Learning Strategy
Action (Method)
Perform Method
(Either transfer knowledge or perform learning step)
Changes to Knowledge Sources
intermediate status information

Figure 4: Architecture of the multistrategy learning module.

```
//  Definition of class Introspector
class  Introspector  {
 private:
  int  tableStrategiesCondition[MAX_PARAMETERS];
                                  //  Solution    (1 - correct, 0 - incorrect or no solution)
                                  //  Heuristics   (1 - yes, 0 - no)
                                  //  Associations (1 - yes, 0 - no)
                                  //  Expertinput  (1 - yes, 0 - no)
  int  qAGoal;                       //  True if learning goal requires question-answer session
 public:
    Introspector(void);               //  Default constructor
    TraceCollection    traces;         //  Collection of traces for postponement
    void  fillTable(void);             //  Method to fill the tableStrategiesCondition
    void  executeAppropriateStrategy(void);
                                  //  Method to execute  appropriate strategy based on table
    void  strategyUnH(void);            //  Strategy for completely unsupervised learning
                                  //       when heuristics used for problem solving
    void  strategySH(void);             //  Strategy for both supervised and unsupervised learning
                                  //       when heuristics used for problem solving
    void  strategySHP(void);            //  strategy for both supervised and unsupervised learning
                                  //       when heuristics used for problem solving
                                  //       and learning goal needs to be suspended
    void  strategyDelS(void);           //  Strategy to delete obsolete associative knowledge
                                  //       through supervisory input
    void  strategyDelUn(char*      inputStr);
                                  //  Strategy to delete obsolete associative knowledge
                                  //       through unsupervised reasoning
    void  learningMethod(void);          //  The learning control method
    void  reinitializeTable(void);
                                  //  Reinitializes tableStrategiesCond
    int  solution(void);               //  Determines if solution is correct by performing tests on
                                  //       the world and comparing reasoning trace
    int  heuristics(void);              //  Determines if reasoning process involved heuristic knowledge
                                  //       by searching through reasoning trace
    int  associations(void);            //  Determines if reasoning trace involved associated knowledge
                                  //       by searching through reasoning trace
    int  expertInput(void);             //  Determines if expert input is available
                                  //       by interacting with user
    void  setQAGoal   (int  val)        //  Creates learning goal for question-answer session
    int  getQAGoal    (void)           //  Returns learning goal for question-answer session
    void  preQA(char*    inString);      //  Pre questio-answer steps
    void  qA(void);                   //  Asks question and gets answer

    void  appendTrace    (void);        //  Utility methods
    void  displayTrace    (void);
    void  displayTrace    (char*   input);

    void  removeTrace    (void);
    void  removeTrace    (char*   input);
    Boolean   inputTrackTrace(char*      input);

};
```

Figure 5: Implementation of introspector as a class in C++.

missing or incorrect knowledge) or when the system ascertains that the problem solving process can be made more efficient. When the introspector determines that something is to be learned, it performs the task of hypothesis generation which results in a hypothesis or explanation of the system's reasoning process. This identifies the knowledge structure to be learned or improved. Hypothesis generation is interleaved with the hypothesis verification process. During this process, the introspector uses information gathered from both the verification actions and tests on the world to test and verify candidate hypotheses. Finally, based on what needs to be learned, an appropriate learning strategy is triggered, which results in the modification of existing knowledge in the problem solving system. The algorithm for introspective multistrategy learning in Meta-TS is shown in table 3.

3.3 Learning strategies

Meta-TS has several strategies for learning associative knowledge for the troubleshooting task, including unsupervised knowledge compilation, supervised learning from an expert, postponement of learning goals, and forgetting invalid associations. Each strategy requires us to make several design decisions; these are discussed below. All the strategies discussed below are fully implemented.

3.3.1 Unsupervised learning

The first strategy is that of unsupervised, incremental inductive learning, which creates an association when the problem solving module arrives at a correct diagnosis using heuristic knowledge. The introspector compiles the heuristic knowledge into an association using a learning method similar to knowledge compilation [Anderson, 1989]. The motivation for this type of learning is performance gain through reduction of the number of intermediate steps when the system encounters a similar problem in the future, although use of this strategy also reinforces correct problem solving sequences.

An example of the unsupervised learning of associations through experience is shown in figure 6. In this example, the ICT ticket reading indicated that the resistor component r22 had failed with a measured reading of 16 ohms. The nominal reading of this component, from the PCB specification, is 20 ohms. The problem solving module reads the symptom (step 1 in the figure). It first tries to find an association that directly maps the observed symptoms into a diagnosis, but fails to find one (step 2). It then finds (step 3) and invokes (step 4) a heuristic that recommends performing the

```
Step 0: Perform troubleshooting and record in Trace Meta-XP, including reasoning steps and
        knowledge (associations or heuristics) used in each step.

Step 1: Analyze Trace Meta-XP to identify "reasoning failures," including incorrect diagnosis,
        inability to create a diagnosis, and correct diagnosis but through inefficient problem solving.

Step 2: If analysis reveals a reasoning failure, then learn:

        Step 2A: Characterize type of reasoning failure
        Step 2B: Use Introspective Meta-XPs encoded in introspector to determine cause of failure
        Step 2C: Use analysis of type and cause of failure to determine what to learn
        Step 2D: Choose appropriate learning algorithm
        Step 2E: Apply learning algorithm
```

Table 3: Algorithm for introspective multistrategy learning in Meta-TS. Note that step 2E is not necessarily performed immediately after 2D; in some cases, it may be performed at a later time (for example, as in the case of the postponement strategy).

troubleshooting action ohming out on r22. The action is performed in step 5, but it finds that r22 is not faulty. Finally, the system outputs the diagnosis that the ICT ticket reading was bogus.

These steps are stored in a Trace Meta-XP, which is analyzed after the troubleshooting is complete. In this example, the introspector performs additional tests on the PCB to determine that the diagnosis is correct. Since this is an experience in which a correct diagnosis was reached through the use of heuristic knowledge in a situation for which no association existed, an Introspective Meta-XP recommends that a new association be learned: "If the ICT ticket indicates that r22 has failed, and the measured reading is slightly lower than the nominal value, then output the diagnosis that the ICT ticket is bogus." This association is installed in the system and is used for future problem solving; it may also be deleted later if it is incorrect or becomes obsolete (e.g., if the problem is fixed).

Several design decisions were made in our implementation of this learning strategy:

- *What is the right time to activate the strategy?* Unsupervised learning takes place at the end of a troubleshooting episode. This strategy is activated when Meta-TS arrives at the right solution using heuristic knowledge alone.

- *When is it useful to form an association?* Meta-TS uses statistical information about the episode (e.g., the number of steps involved in problem solving) and determines if there will be performance gain through the reduction in the number of intermediate steps while troubleshooting a similar board. This information is only used to determine whether learning a new association would speed up the troubleshooting process, and does not ensure that the learned association is "correct."

- *What is the right association to learn?* Consider the situation when the ICT input is I, the intermediate steps are 1, 2, 3, 4, and 5, and the diagnostic result is O. Meta-TS would form an association either between I and O, or between I, the final step (step 5 in this example), and O. Domain knowledge is used to decide between the two alternatives. Our data shows that human operators typically form an association between the input and output without any intermediate steps when the diagnostic result is "Bogus ICT ticket reading." In contrast, when the operator decides to replace a defective part, he or she is conservative and performs either a visual inspection or some other intermediate action to confirm the hypothesis. Meta-TS behaves in a similar manner.

```
Troubleshooting    board  #13

ICT  ticket  information
    _________________________
    These  are  board  faults
    TA  7052  MAX  Processor
    _________________________
    r22  has  failed
    Measured   = 16.000000   ohms
    Nominal    = 20.000000   ohms
    _________________________

Entering   problem   solver

Step  #1
  CONTROL   METHOD:   get  symptom   information    from  ticket
  PRECONDITIONS:     ticket  available
  MET  BY:  input
  RESULT:   r22  has  failed

Step  #2
  CONTROL   METHOD:   find  associations    for  r22
  PRECONDITIONS:     symptom   available
  MET  BY:  r22  has  failed
  RESULT:   no  associations    for  r22

Step  #3
  CONTROL   METHOD:   find  heuristics
  PRECONDITIONS:     symptom   available   AND  no  associations
  MET  BY:  r22  has  failed  AND  no  associations    for  r22
  RESULT:   heuristic-1    found

Step  #4
  HEURISTIC:    apply  heuristic-1
  PRECONDITIONS:     measured   value   is  slightly   lower   than   nominal   value
  MET  BY:  ticket  information
  RESULT:   ohming-out    recommended

Step  #5
  ACTION:   ohming-out    on  r22
  PRECONDITIONS:     symptom   available   AND  action   recommended
  MET  BY:  r22  has  failed  and  ohming-out    recommended
  RESULT:   bogus  ict  ticket

Diagnosis:    bogus  ict  ticket

Entering   learner

Analyzing    Trace  Meta-XP   steps  1-5
  SYMPTOM:   r22  has  failed
  DIAGNOSIS:    bogus  ict  ticket
  OUTCOME:   diagnosis   correct
Using  Introspective    Meta-XP
  GOAL:  learn  association    for  r22
  STRATEGY:   unsupervised    learning   of  associations

Invoking   learning   strategy
Creating   new  association    for  r22

Meta-TS   now  has  associations    for
r243,  u37,  r254,  r200,  r121,  u73,  r22
```

Figure 6: An example of the strategy of unsupervised learning of associations in Meta-TS. Here, Meta-TS has just learned the association that r22 failures often indicate a bogus ticket reading.

Discussion: We observed that human operators used yellow tags ("PostIt notes") to note down a recurring problem, especially when they believe that this information will be useful in the future. This happened when they performed several intermediate steps during troubleshooting, and typically after they had arrived at the diagnostic result. This was the motivation for including this learning strategy, and also the basis for the first two design decisions.

3.3.2 Supervised learning

The second learning strategy creates a new association through supervisory input. This strategy is triggered when the system arrives at an incorrect solution using heuristic and/or associative knowledge. The system attempts to acquire a correct associative knowledge from a skilled troubleshooter (the "supervisor"). This mechanism is similar to the interactive transfer of expertise in TEIRESIAS [Davis, 1979]. However, the knowledge learned in our system is not in the form of production rules, but in the form of frames and slots for association records.

An example of the supervised learning of associations through experience is shown in figure 7. In this example, the ICT ticket reading indicates that the resistor component r24 has failed, the measured reading of 21.2 ohms being much higher than the nominal value of 10 ohms. There are no known associations for this problem, so the system applies a heuristic that recommends ohming out on r24. In this example, ohming out confirms that the ticket reading was correct. Another heuristic recommends a simple visual inspection of the PCB, which shows that r24 is missing from this PCB. This is output as the diagnosis from the problem solving module. The introspector in the learning module finds that the diagnosis is not correct; in this case, there is a missing IC component, u37, that is responsible for the problematic ICT ticket reading. The expert supervisor suggests that a new association be formed that, for this input, recommends performing a visual inspection on u37. This association is learned and installed for future use.

Several design decisions were made in our implementation of this learning strategy:

- *What is the right time to activate the learning strategy?* This strategy is activated at the end of the troubleshooting episode when the system arrives at an incorrect solution or is unable to make any inference based on the information available to it.
- *What is the structure of the supervisory input?* The structure of the desired supervisory input is determined by the manner in which the associative knowledge is stored in the

```
Troubleshooting board #2

ICT ticket information
    ------------------------
    These are board faults
    TA 7052 MAX Processor
    ------------------------
    r24 has failed
    Measured  = 21.200001 ohms
    Nominal   = 10.000000 ohms
    ------------------------

Entering problem solver

Step #1
  CONTROL METHOD: get symptom information from ticket
  PRECONDITIONS: ticket available
  MET BY: input
  RESULT: r24 has failed

Step #2
  CONTROL METHOD: find associations for r24
  PRECONDITIONS: symptom available
  MET BY: r24 has failed
  RESULT: no associations for r24

Step #3
  CONTROL METHOD: find heuristics
  PRECONDITIONS: symptom available AND no associations
  MET BY: r24 has failed AND no associations for r24
  RESULT: heuristic-3 found

Step #4
  HEURISTIC: apply heuristic-3
  PRECONDITIONS: measured value is much higher than nominal value
  MET BY: ticket information
  RESULT: ohming-out and visual-inspection recommended

Step #5
  ACTION: ohming-out on r24
  PRECONDITIONS: symptom available AND action recommended
  MET BY: r24 has failed and ohming-out recommended
  RESULT: ticket reading verified

Step #6
  ACTION: visual-inspection on r24
  PRECONDITIONS: symptom available AND action recommended
  MET BY: r24 has failed and visual-inspection recommended
  RESULT: r24 is missing

Diagnosis: missing part, r24 is missing

Entering learner

Analyzing Trace Meta-XP steps 1-6
  SYMPTOM: r24 has failed
  DIAGNOSIS: missing part, r24 is missing
  OUTCOME: diagnosis incorrect, u37 is defective
Using Introspective Meta-XP
  GOAL: learn association for u37
  STRATEGY: supervised learning of associations

Invoking learning strategy

Is expert input available for this episode?  >> yes
  Enter left-hand-side of association:  >> u37
  Select right-hand-side of association:
  a. Bogus ICT ticket
  b. Replace defective part
  c. Perform visual inspection followed by diagnosis
  d. Perform lifted leg procedure followed by diagnosis
  >> c

Creating new association for u37

Meta-TS now has associations for
r243, u37
```

Figure 7: An example of the strategy of supervised learning of associations in Meta-TS. Italics indicate user input during this episode.

system. In contrast, the conversation between an expert and novice troubleshooter is not so structured. Since the relevant information transmitted between them is domain- and task-oriented, however, that structure is exploited in the dialogs used by Meta-TS. While the current implementation of this learning strategy does not model the full richness of a troubleshooter's interactions with an expert, the more structured interaction allows an objective evaluation of the model. The user interaction in the current implementation of the system is very simple since that was not the focus of our research; however, it would be relatively easy to include a more sophisticated dialog system if desired.

- *How can the system reason about the validity of the expert input?* This is an open question for learning systems in general. However, for our purposes, the input from the expert troubleshooter can be assumed to be correct. Meta-TS does not critically examine whether the input given by the expert is correct; it directly takes the associative knowledge input by the expert and adds it to its knowledge base.

Discussion: Novice operators ask expert troubleshooters such as engineers or highly trained technicians when they have problems in their task. We use the expert-novice metaphor for the supervisor-system interaction. The system learns the knowledge input by the supervisor (as do novice troubleshooters). The interaction between Meta-TS and the expert is capable of gathering the relevant associative knowledge. However, the actual mode of communication does not reflect expert-novice interaction in the real world. For example, Meta-TS currently does not model apprenticeship relationships in troubleshooting (see, for example, [Redmond, 1992]).

The improvement in system performance from this learning strategy depends on the quality and validity of the expert input. The new knowledge is subject to change, depending on the future episodes encountered by the system. If the new knowledge obtained from supervisory input is found to be reliable in a number of future instances, the confidence in the gained knowledge is increased. However, if the new knowledge is incorrect, it is deleted over time (see section 3.3.4). Thus, the transfer of knowledge is immediate but the "sustainability" of the knowledge depends on the use of the gained knowledge.

3.3.3 Postponement

A third learning strategy is that of postponement [Ram, 1991]. This strategy is triggered when the system is unable to get immediate input from a skilled troubleshooter. The system posts a learning goal [Ram, 1991; Ram and Hunter, 1992; Ram and Leake, in press], keeps track of the reasoning trace for the particular problem solving episode, and asks questions at a later time to gather appropriate associative knowledge. Postponement takes place when there is no supervisory input at the end of a troubleshooting episode. The learning goal and the trace of the troubleshooting episode are stored in the introspector. Suspended learning goals can be satisfied both through supervised or unsupervised methods at a later time.

At the beginning of a new troubleshooting episode, the introspector checks whether the reasoning trace associated with any suspended learning goal is based on an input problem that is similar to the current problem. Similarity is determined based on the fault type indicated on the ICT ticket and the difference between the nominal and measured readings. If one or more matching learning goals are found and an expert is available, the introspector triggers a question-and-answer session by presenting the information it has on the past episodes. Details of the episodes are presented only if the supervisor desires to look at it. If expert input is obtained, new associative knowledge is added to the system and the resolved learning goals are deleted along with the associated reasoning traces. The system then continues to solve the current problem using the new associative knowledge.

If no expert input is available, the introspector tries to solve the current problem. If it succeeds, the learning goals that matched this problem are automatically satisfied without supervisory input. As before, these goals and associated reasoning traces are deleted since the system is now capable of solving those problems. The system is also capable of solving similar problems in the future with the newly formed associative knowledge.

Again, several design decisions were made in our implementation of this learning strategy:

- *What is the appropriate time for question-answer sessions?* A question-answer session takes place either at the end of a troubleshooting episode or at the beginning of a new episode. Question-answer sessions are not needed for learning goals which become redundant when new associative knowledge is learned without user input. There are, of course, several other factors involved in deciding when to ask a question, including sociological factors such as the personalities of and interpersonal interactions between the troubleshooter and the expert technician; these are outside the focus of our model.

- *How should the suspended question be presented?* Meta-TS uses context-sensitive presentation of information. When the user is asked for input in a situation which matches a similar situation that is associated with a learning goal suspended from a prior episode, the information in the reasoning traces leading to that learning goal is presented to provide a context for the dialog. Using the principle of progressive disclosure, the user can ask to examine more details.

- *When are learning goals active?* Learning goals are always "active" in the sense that any problem solving episode or question-answer session could contain the information sought by a prior learning goal; however, learning goals are not actively pursued by the system until the desired information is available in the available input, at which time the algorithm that carries out the learning is executed.

Discussion: Novice operators seek input from the expert supervisor when they are unable to find the solution to a problem. Operators may ask for input when a similar new problem is encountered. In the assembly plant, operators keep PCBs that have successfully been diagnosed in a bin, and often re-examine these PCBs to find information relevant to the current problem. Undiagnosed PCBs may also be stored and retrieved later for re-analysis, which corresponds to the deferment of a learning goal until a later opportunity to get the appropriate information is encountered. The design decision to present prior reasoning traces to the expert is intended to facilitate user interaction; although operators can often recall what they did in earlier situations, it is arguable whether they remember all the details of the entire troubleshooting process for the earlier situations.

3.3.4 Forgetting

Two additional learning strategies delete associative knowledge when it is no longer valid. These strategies are primarily targetted at the brittleness problem that is encountered when the manufacturing process is changed and existing associations are rendered obsolete. The first strategy uses expert input to delete associations, and is invoked at the end of every problem solving episode. The system queries the supervisor to determine whether any associations used in the reasoning trace of that episode should be deleted. If the supervisor has knowledge about, for example, a process change and the system dynamics has resulted in an association becoming obsolete, that information can be input to Meta-TS. This strategy works quite well in general, although it is, of course, dependent on the availability and quality of user input.

The second deletion strategy is unsupervised and does not require user input. This strategy is selected when Meta-TS arrives at an incorrect solution (as determined through additional tests on the PCB or through expert input) and the reasoning trace shows that a single association was used in arriving at the solution. Since heuristic knowledge in this task domain tends to be relatively stable, an incorrect diagnosis involving several heuristics and a single association is blamed on the association. The introspector tracks down this association and deletes it. The current implementation of this strategy cannot deal with situations in which more than one association is used; such situations require assigning blame to the particular association that was at fault.

Several design decisions were made in our implementation of this learning strategy:

- *Under what conditions should an association be deleted?* When the expert troubleshooter indicates that an association needs to be deleted, Meta-TS follows the supervisory input. In the unsupervised mechanism, the system behaves conservatively in the sense that a piece of associative knowledge is deleted only if the diagnostic result is incorrect and only one association was involved in the problem solving process. In the current implementation, the system deletes an association if it leads to a single incorrect diagnosis; while not a general solution to the problem of determining when a piece of knowledge is no longer valid, this method is reasonable in our task domain given the highly dynamic nature of the manufacturing process. Another learning strategy (not currently implemented) would be to make the association more specific so as to exclude the current situation.
- *What is the right time to activate the strategies?* Deletion of existing associative knowledge in Meta-TS takes place at the end of a troubleshooting episode. At this point, the system has available to it the trace of its reasoning process and also information about the correctness of its diagnostic result. Both are required in order to identify and delete incorrect knowledge.

Discussion: When the manufacturing process changes, it impacts the quality of the boards produced, the types of malfunctions that can occur, and consequently the operator troubleshooting. For example, let us assume that r243 is a known defective part, say, due to a poor quality vendor. When the vendor is changed, the part r243 may no longer be defective. Typically, this information is communicated from the manufacturing process line or when the operator recognizes the change in the situation due to a failure of the troubleshooting process. The first situation corresponds to the supervisory input case, and the second to the unsupervised case. It is arguable whether human operators can "forget" an association instantaneously; however, trained operators often stop using an obsolete association even if they do not actually "forget" it. The cognitive plausibility of various forgetting mechanisms is still an open research issue, although the methods implemented in Meta-TS are effective in dealing with the particular task at hand.

4 Evaluation

Meta-TS has been evaluated both qualitatively and quantitatively. We were interested both in comparing the results to the human data, as well as evaluating it as a machine learning system. We evaluated the system using 42 actual problem solving episodes gathered at the plant over a 2-month period [Cohen, 1990]. The problems dealt with various types of resistor failures and are representative of the types of problems encountered over the 2-month period. To evaluate the learning methods, we tested the following five conditions on the 42 test problems.

- H (hand-coded): The original non-learning system with hand-coded associations. This condition represents a troubleshooting system that has been hand-designed by an expert researcher, and is useful as a benchmark in determining the strengths and limitations of the learning strategies.
- NL (no learning): The system with all associations removed and learning turned off. This condition represents a base case against which to evaluate the efficacy of the learning strategies; it uses only heuristic knowledge.
- L (learning): The system with all associations removed and learning turned on. This is the basic Meta-TS system with no prior experience.
- L42: The system with all associations removed, then trained it on the 42 test problems with learning turned on. The system was then evaluated by re-running it on the same 42 problems. This condition was intended to validate the learning strategies in Meta-TS by ensuring that they learned the knowledge required to solve the problems.
- L20: The system with all associations removed, then trained on 20 randomly generated training problems with learning turned on. The problems can be classified as easy, medium, and hard, based on degree of difficulty as measured using the number of intermediate steps in the troubleshooting process. We generated 20 random problems with the probabilities that the problem generated was easy, medium or hard set to 0.6, 0.2 and 0.2, respectively.

The randomly generated training set is representative of the problems a human operator encounters over about a month at the job, both in terms of number and degree of difficulty. The problems varied from 42 test problems in various ways.

Each of these conditions were evaluated quantitatively for speed and accuracy on the 42 test problems, and also qualitatively by examining the content of the learned knowledge and details of the solution process. For supervised learning strategies, we provided "expert" input to the system based on what was appropriate to the input problem and domain experience.

4.1 Quantitative evaluation

Two quantitative performance measures were used: the accuracy of the diagnostic result, and the speed (measured by the number of intermediate problem solving steps) of arriving at the diagnosis. Figures 8 through 12 illustrate the system performance over the 42 problems for the H, NL, L, L42 and L20 conditions.

Diagnostic accuracy: Figure 8 shows the cumulative accuracy of the system for the various conditions. The H condition arrived at the correct diagnosis in 86% of the 42 problems. The L42 condition arrived at the correct diagnosis in 81% of the problems. The values for the L20, L, and NL conditions were 79%, 76%, and 71% respectively. The graphs illustrate both these final accuracy figures, as well as the improvement of the system with experience.

Figures 9 and 10 compare the accuracy of the learning conditions relative to that of the hand-coded condition and non-learning conditions, relatively. By measuring the ratio, we compensate for differences in the intrinsic difficulty of the individual problems. Again, the graphs illustrate both the final result as well as improvement with experience. The ratio of the L42 condition to that of the H condition is about 0.94; for L20 and L conditions, the ratios are 0.92 and 0.89, respectively. As compared with the NL condition, the L42 condition is about 1.14 times more accurate; for L20 and L, the ratios are 1.11 and 1.07, respectively.

Speed of problem solving: Figures 11 and 12 compare the speed of the solution process (measured by the number of intermediate steps) with the various learning conditions relative to the hand-coded and non-learning condition, respectively. The L20 and L42 conditions consistently

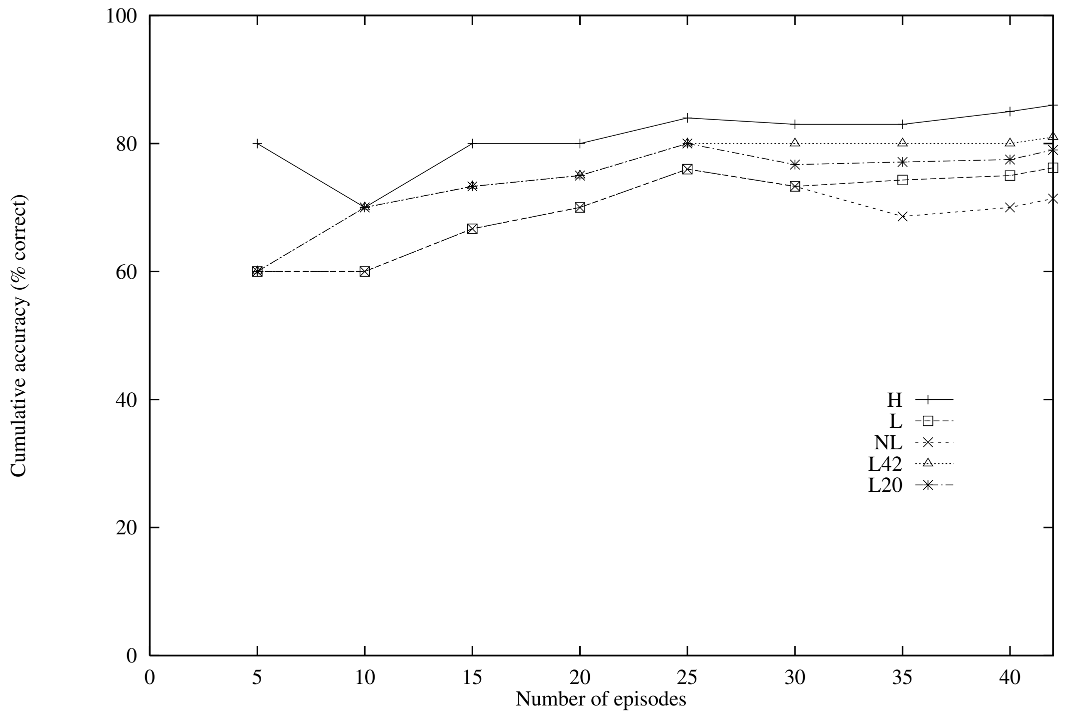

Figure 8: Cumulative diagnostic accuracy

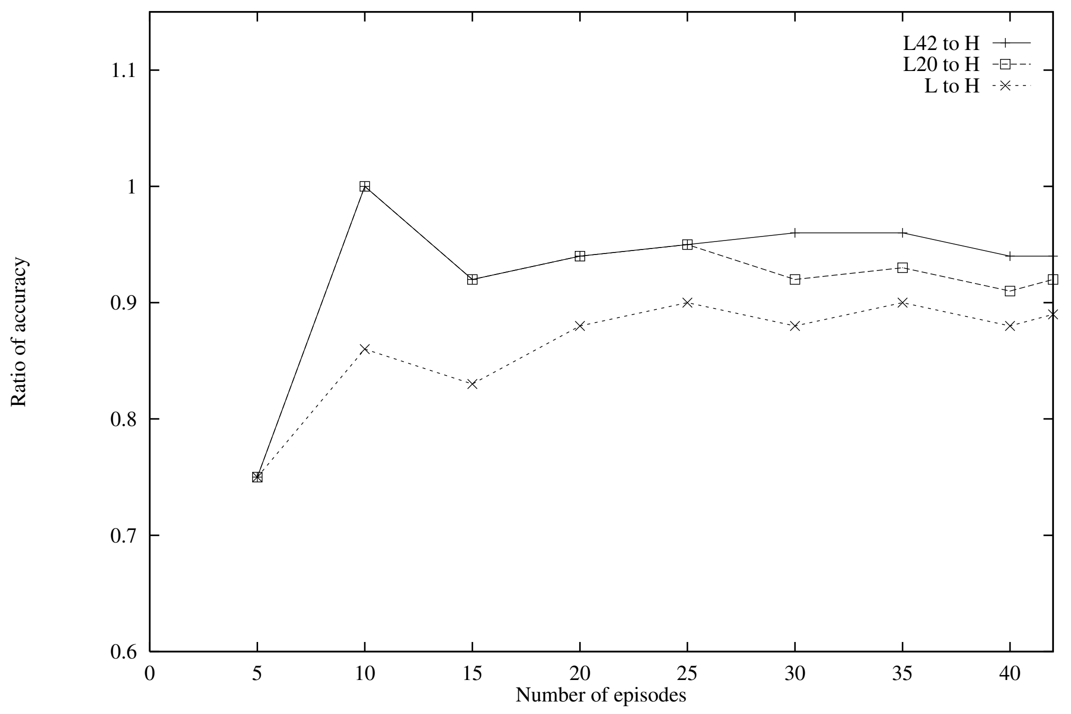

Figure 9: Ratio of diagnostic accuracy of learning conditions to hand-coded associations

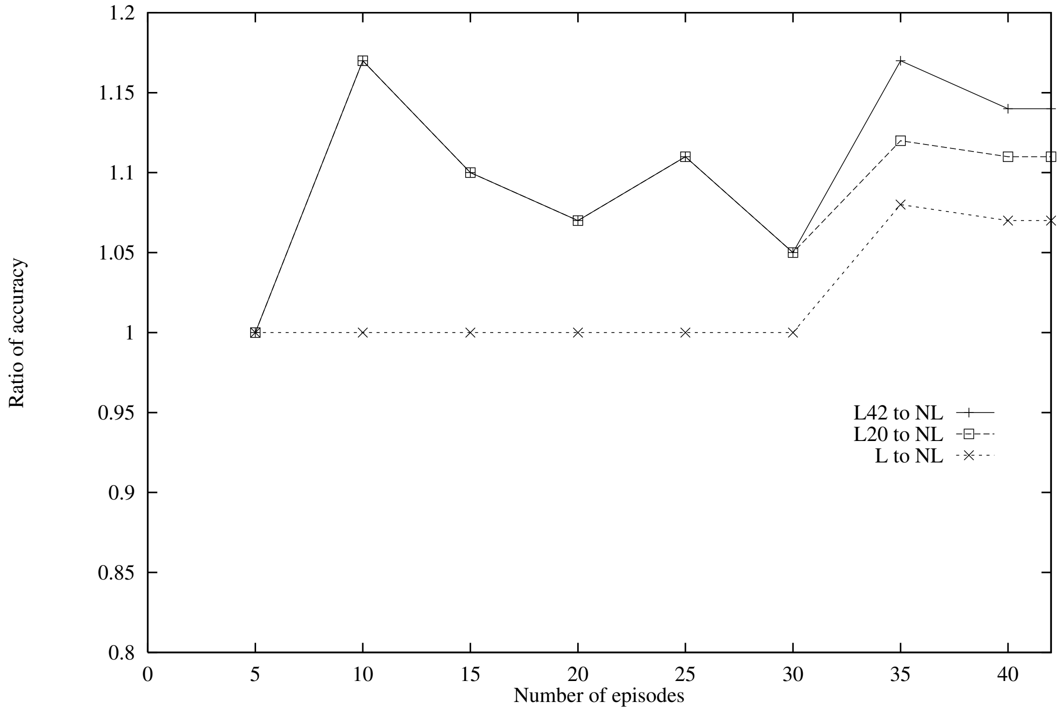

Figure 10: Ratio of diagnostic accuracy of learning conditions to non-learning condition

arrive at the diagnostic result faster than the H condition. The L condition takes about 20 problem episodes to reach the same speed as that of the H condition and then consistently arrives at the diagnostic result faster than the H condition. At the end of the 42 problem episodes, the ratios of the learning conditions to the hand-coded conditions are: 1.52 (L42 to H), 1.24 (L20 to H), and 1.06 (L to H). In comparison to the non-learning version of the program, all the three learning conditions, L42, L20, and L, consistently arrived at the diagnostic result faster than the NL condition. At the end of the 42 problem episodes, the ratios of the learning conditions to the hand-coded conditions are: 1.75 (L42 to H), 1.41 (L20 to H), and 1.20 (L to H).

Discussion: The results of the quantitative evaluation can be summarized as follows. The multistrategy learning module in Meta-TS clearly contributes to enhanced system performance in the troubleshooting task. In comparison with the non-learning system with no hand-coded associations, the associative knowledge learned by Meta-TS increases the accuracy of the diagnostic result and speeds up the problem solving process. The performance of Meta-TS further increases when it is trained on similar problems before it is applied to novel problems. The associative knowledge learned by Meta-TS enables it to arrive at the same solution as that of the system with the hand-coded associative knowledge between 89% and 94% of the time.

Although Meta-TS is faster than the hand-coded version, it was also seen that Meta-TS with the various learning strategies did not outperform the system with the hand-coded associations in terms of the accuracy of diagnostic result. We hypothesize that it may be due to two reasons. First, in order not to spoon-feed the system and possibly invalidate the results, the supervisory input given to the system throughout the evaluation process was kept very minimal. Thus, the expert input to the system for either the 20 or 42 problem solving episodes may not have enabled Meta-TS to obtain all the associations that an operator in the plant obtains over a period of several months of task performance.[3] Second, the system as currently implemented may not contain all the learning strategies that a human operator uses. However, given the learning architecture used in Meta-TS, it is relatively straightforward to incorporate additional learning strategies, once identified, in the system.

[3]In the experiments discussed above, Meta-TS asked for supervisory input in 9 out of the 42 episodes, and supervisory input was provided in 5 of those episodes (i.e., about half the time). Thus, supervisory input was actually obtained in about 12% of the troubleshooting episodes. While this is reasonable for the purposes of evaluating our model, the actual percentages vary according to factors beyond the scope of our model, such as the nature of the professional relationship between the novice troubleshooter and the expert supervisor.

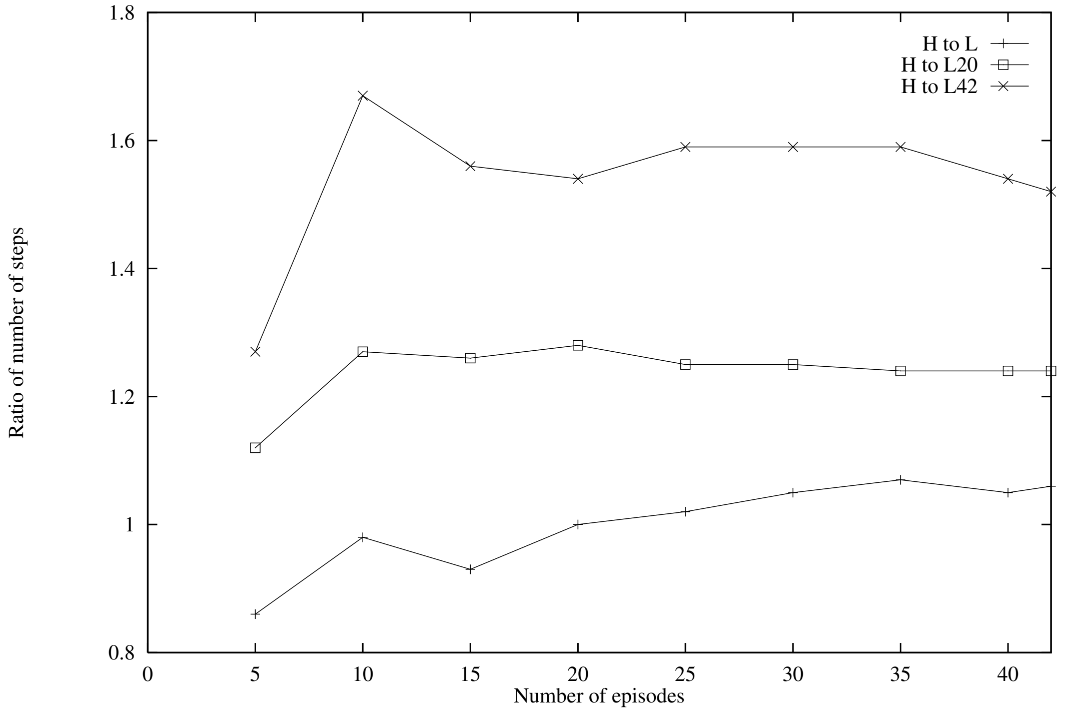

Figure 11: Ratio of problem solving speed of learning conditions to hand-coded associations

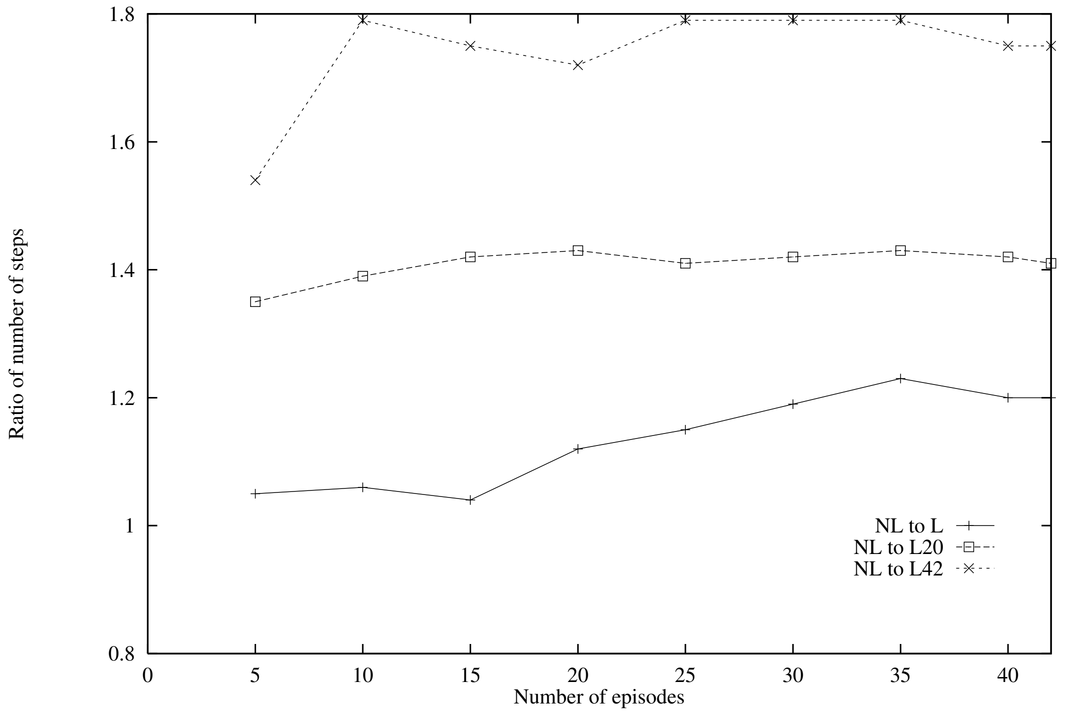

Figure 12: Ratio of problem solving speed of learning conditions to non-learning condition

4.2 Qualitative evaluation

We also evaluated Meta-TS using various qualitative metrics. We compared the learned associations with the hand-coded associations, the solution process of a human operator to that of Meta-TS on the same problems, and the methods and knowledge used by Meta-TS to troubleshoot and learn to those used by human operators. The results are as follows.

Quality of the learned associative knowledge: We compared the associative knowledge learned by Meta-TS while troubleshooting the 42 test boards to the hand-coded associations in the original problem solving system. Meta-TS learned 33% of the hand-coded associations. It was unable to learn some of the hand-coded associations as it did not encounter them in the training or test problem set. (Recall that the hand-coded associations were based on over 300 problem solving episodes.) Meta-TS also learned other associations that did not correspond to the hand-coded ones which enabled it to perform better in terms of the speed of the solution process.

Comparison of the solution process: We compared the process of arriving at a solution in Meta-TS and operator troubleshooting processes from the verbal protocols. The L20 condition was used in this comparison as it best represents a fairly trained operator because of the training input discussed earlier. We divided the problems into two sets. Difficult problems included problems in which Meta-TS was unable to arrive at the correct solution or those which required several intermediate problem solving steps; in about 50% of these problems, human operators also spent a considerable time in troubleshooting. The remaining problems were considered easy for Meta-TS; in about 80% of these, human operators also arrived at the correct solution fairly quickly.

Comparison of troubleshooting knowledge and learning processes: As discussed earlier, human operators rely predominantly on shallow reasoning methods using heuristic and context-sensitive associative knowledge in this task domain. This is modelled through the use of heuristic and associative knowledge in the troubleshooting model. Furthermore, humans operators learn associative knowledge through experience in the task by several means. They may obtain input from expert troubleshooters. They may also notice recurring instances of a problem and may then form an association between the input and the diagnostic result. In some situations, when immediate input from an expert is not available, they may place the PCB with the ICT reading aside and then attempt to obtain supervisory input at a later time when they encounter a similar problem. All these means of learning associative knowledge by human operators are reflected by the various learning strategies in Meta-TS, as discussed earlier. The strategies are integrated through the introspective learning architecture of the system.

4.3 Generality of the model

In addition to evaluating Meta-TS itself, we also need to evaluate the generality and flexibility of the underlying model of introspective multistrategy learning. We are performing another case study in a task domain that is very different from the one discussed in this article. The Meta-AQUA system, presented in Ram and Cox [in press], uses "deep" causal knowledge to understand natural language stories. The performance task in this system that of causal and motivational analysis of conceptual input in order to infer coherence-creating structures that tie the input together. Meta-AQUA is an introspective multistrategy learning system that improves its ability to understand stories consisting of sequences of descriptions of states and actions performed by characters in the real world. The system is based on the AQUA system [Ram, 1991; Ram, 1993], which is a computational model of an active reader. Meta-AQUA uses the same theory of introspective multistrategy learning to allow the system to recover from, and learn from, several types of reasoning failures through an introspective analysis of its performance on the story understanding task.

In both AQUA and Meta-AQUA, reading is viewed as an active, goal-driven process in which the reasoner focusses attention on what it needs to know and attempts to learn by pursuing its goals to acquire information [Ram, 1991]. Such a system models the hypothetical metacognitive reader discussed by Weinert [1987], who "perceives a gap in his knowledge, ... attempt[s] to take notes on the relevant information, to understand it," undertakes "learning activities from a written text," examines "how his assessment of his own knowledge structures compares with his expectations about the demands" of an uncoming performance task, and can tell us about his "preferred learning strategies, and his evaluation of his own situation and the possible consequences" [p. 7]. While reasoning in this task domain is very different from the often shallow diagnostic processes used in assembly line manufacturing, and the two use very different kinds of knowledge, it is possible to use the same model of introspective multistrategy learning in both task domains [Ram, Cox, and Narayanan , in press]. Although further details of Meta-AQUA are outside the scope of this article, we introduce the system here as further computational evidence of the generality of our approach.

4.4 Limitations of the model

While we have achieved a reasonable degree of success in modelling human troubleshooters as they learn and gain experience on an assembly line, our model also has several limitations. Some of these limitations are due to the introspective multistrategy learning theory, which is discussed in more detail in the following section. Here, we discuss limitations in our use of the theory as a computational model of human troubleshooting, including limitations arising from the computational framework used to develop Meta-TS, and limitations due to the current implementation of the Meta-TS program.

Implementational limitations are, perhaps, the least important. For example, our current implementation of the method for interactive transfer of expertise during supervised learning is very simple. We were interested in the integration of multiple learning methods into a single system and not so much in developing new learning algorithms. Similarly, the implementation of forgetting simply involves deletion of an association; clearly, human forgetting is a much more complex process. Other such simplifications have been pointed out in the preceding technical discussion. It is interesting to note that Meta-TS can model many aspects of the human data even with these simplified learning strategies; furthermore, should it be necessary to model additional details of a particular learning scenario or to add new learning scenarios to the model, the framework used in Meta-TS makes it relatively straightforward to do so.

Meta-TS is also limited in certain ways as a computational model of human troubleshooting. Our model focusses on ICT troubleshooting operators who routinely work on testing and repair, and does not model technicians or engineers who are, for example, called in to help with this task on certain occasions, such as when a very difficult problem is encountered. Although expert technicians and engineers may also rely on associative and heuristic knowledge similar to that observed in our study, they may also use other kinds of knowledge, such as topographic models or causal knowledge. For example, Hale [1992] shows that humans use both weak causal heuristics and domain-specific knowledge in learning symptom-fault associations in causal domains. As before, while the Meta-TS framework permits extension of the model, the current model does not represent these kinds of knowledge.

Another limitation of the current model is the simplified view of the troubleshooter's interaction with the environment. This interaction not only includes expert-novice interaction in supervised learning situations, but also includes interaction with the equipment and artifacts in the environment that the troubleshooter is situated in. In particular, our model focusses on cognitive processing and not on situated interactions; while the former is important, the relationship between the two is an important issue for future research.

Finally, while the learning strategies used in Meta-TS are similar to those used by a typical "trained" operator, and the overall learning behavior of Meta-TS is also comparable with that of a human operator, our analysis does not provide a detailed comparison with human thought processes on individual problems. In particular, on a given set of problems, we have neither shown that an individual human operator formulates the particular reasoning traces that Meta-TS does, nor that he or she selects the particular learning strategies that Meta-TS does on each problem in that set. Such a comparison is extremely difficult since the specifics of a reasoning trace, and the corresponding choice of a learning strategy, depend on the domain knowledge and level of expertise of the troubleshooter, the prior problems encountered, the availability of an human expert, and other details. Furthermore, it is unclear how one could obtain protocols of human troubleshooters that specified their reasoning traces or their strategy selection decisions in sufficient detail to permit direct comparison on individual problem solving episodes at the level of granularity of the computational model.[4] Thus, Meta-TS should be viewed as a model of a typical troubleshooting operator in a typical assembly line environment, and not as a detailed model of a specific individual operator solving a specific set of problems.

5 Discussion and related research

We have presented a computational framework for introspective multistrategy learning, which is a deliberative or strategic learning process in which a reasoner introspects about its own performance to decide what to learn and how to learn it. The reasoner introspects about its own performance on a reasoning task, assigns credit or blame for its performance, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies. In this article, we have presented a model of human troubleshooting based on this framework, focussing in particular on the learning

[4] A recent article by Pirolli and Recker [in press] presents a methodology used to obtain and analyze human protocols of metacognitive behavior during troubleshooting LISP programs and the consequent effects on learning. Although the learning strategies of concern to them are at a different level of granularity than the ones modelled in Meta-TS, their methods and results are encouraging and not incommensurable with those presented here.

aspects of the model. The model is implemented in a computer program which models human troubleshooters at an operational electronics assembly plan and also provides a case study in the use of the computational framework for the design of multistrategy machine learning systems.

Diagnostic problem solving has been studied by several researchers in cognitive science, artificial intelligence, psychology, and human-machine systems engineering. Specifically, there has been much work on troubleshooting in real-world domains, including that of Bereiter and Miller [1989] in computer-controlled automotive manufacturing, Govindaraj and Su [1988] in marine power plants, Katz and Anderson [1987] in program debugging, Kuipers and Kassirer [1984] in medicine, and Rasmussen [1984] in industrial process control. Much of this work is based on studies of human problem solving. Rouse and Hunt [1984] discuss various models of operator troubleshooting based on experimental studies in simulated fault diagnosis tasks and present implications for training and aiding operators in these tasks. Research in artificial intelligence has resulted in computational models of knowledge-based diagnosis (e.g., [Chandrasekaran, 1988]) and qualitative reasoning (e.g., [deKleer and Williams, 1987]).

A detailed review of research in human troubleshooting and diagnostic problem solving is outside the focus of this article, which is concerned with issues in learning and introspection. In the remainder of this section, we will summarize issues in multistrategy learning and discuss related research in the artificial intelligence and metacognition literatures.

5.1 Multistrategy learning

The basic idea underlying the learning model in Meta-TS is that of introspective multistrategy learning. Previous research on learning in machine learning and cognitive science has resulted in the development of several learning algorithms for different situations, including the algorithms used in Meta-TS. In order to integrate these algorithms into a single multistrategy system, it is necessary to develop methods by which the system can make its own decisions concerning which learning strategies to use in a given circumstance. Often, knowledge about applicability conditions and utility of learning strategies is implicit in the procedures that implement the strategy; this further complicates the strategy selection problem. Our solution to this problem is to represent knowledge of learning strategies and applicability conditions for these strategies explicitly in the system itself.

As discussed earlier, several approaches to multistrategy learning have been proposed, including approaches based on strategy selection models, toolbox models, cascade models, and single mechanism models. Examples of each of these approaches were presented in section 1.2. Our approach is an example of a strategy selection model. To develop a computer program that can deal with the complexities of real-world troubleshooting, the system must deal with an incomplete world model, dynamic changes in the world which renders part of the world model obsolete, and multiple forms of knowledge (much of it shallow). This requires the integration of multiple learning methods (inductive, analytical, and interactive) in both supervised and unsupervised situations. Our experience with the Meta-TS system shows that a strategy selection architecture can deal effectively with such problems. Furthermore, our approach provides a general framework for integrating multiple learning methods. The learning strategies are not dependent on the domain, but are however dependent on the types of knowledge used in the performance task.

The Soar system [Laird, Rosenbloom, and Newell, 1986] takes a different approach to learning, which uses a single learning mechanism (chunking) rather than multiple learning strategies. Instead of explicit representations of different problem solving and learning methods and explicit selection between them, Soar is based on "weak methods" (universal subgoaling and chunking) from which higher-level strategies emerge. Our methodological stance is to develop an explicit theory of the different types of reasoning and learning that the system is to perform. We wish to understand the nature of various learning methods; the kinds of situations that these methods can deal with; the kinds of knowledge that can be learned with them; their limitations; and so on. Although it is possible that a single underlying mechanism might be able to implement all these methods, it is still important to identify and study the methods themselves, particularly in developing computational models of human learning in which behaviors corresponding to these learning methods are exhibited.

5.2 Artificial intelligence and meta-reasoning

There are several fundamental problems to be solved before we can build intelligent systems capable of general multistrategy learning, including: determining the cause of a reasoning failure (blame assignment), deciding what to learn (learning goal formulation), and selecting the best learning strategies to pursue these learning goals (strategy selection). We claim that a general multistrategy learning system that can determine its own learning goals and learn using multiple learning strategies requires the ability to reflect or introspect about its own internals. Pollock

[1989] distinguishes between knowledge about the facts that one knows and knowledge about one's motivations and processes. Introspective multistrategy learning is based on the both kinds of meta-knowledge; we argue that introspective access to explicit representations of knowledge and of reasoning processes is essential in making decisions about what and how to learn.

One form of introspection that has been implemented in many systems is the use of reasoning traces to represent problem solving performance; an early example of this approach was Sussman's [1975] HACKER program. Reasoning trace information has primarily been used for blame assignment (e.g., [Birnbaum, Collins, Freed, and Krulwich, 1990]) and for speedup learning (e.g., [Mitchell, Keller, and Kedar-Cabelli, 1986]). In addition, we propose that such information, suitably augmented with the kinds of knowledge represented in our Introspective Meta-XP structures, can be used as the basis for the selection of learning strategies in a multistrategy learning system.

In this approach, information about classes of learning situations, and learning strategies for each type of learning situation, is represented explicitly in the system. In addition to the world model that describes its domain, the reasoning system has access to meta-models describing its reasoning and learning processes, the knowledge that this reasoning is based on, the indices used to organize and retrieve this knowledge, and the conditions under which different reasoning and learning strategies are useful. A meta-model is used to represent the system's reasoning during a performance task, the decisions it took while performing the reasoning, and the results of the reasoning. As shown in this article, this knowledge can then be used to guide multistrategy learning using introspective analysis to support the strategy selection process.

Many research projects in AI have demonstrated the advantages of representing knowledge about the world in a declarative manner. Similarly, our research shows that declarative knowledge about reasoning can be beneficial. The approach is novel because it allows strategy selection systems to reason about themselves and make decisions that would normally be hard-coded into their programs by the designer, adding considerably to the power of such systems. Meta-reasoning has been shown to be useful in planning and understanding systems (e.g., [Stefik, 1981; Wilensky, 1984]). Our research shows that meta-reasoning is useful in multistrategy learning as well. To realize this ability, our model incorporates algorithms for learning and introspection, as well as representational methods using which a system can represent and reason about its meta-models.

From the artificial intelligence point of view, our approach is similar to other approaches based on "reasoning traces" (e.g., [Carbonell, 1986; Minton, 1988]) or "justification structures" (e.g., [Birnbaum, Collins, Freed, and Krulwich, 1990; deKleer, Doyle, Steele, and Sussman, 1977;

Doyle, 1979]), and to other approaches that use characterizations of reasoning failures for blame assignment and/or multistrategy learning (e.g., [Mooney and Ourston, 1991; Park and Wilkins, 1990; Stroulia and Goel, 1992]). A major difference between our approach and these approaches is our use of explicit representational structures (Introspective Meta-XPs) to represent classes of learning situations along with the types of learning needed in those situations, a type of knowledge that is crucial in multistrategy learning systems. Other types of knowledge may also be important in multistrategy learning systems. For example, Pazzani's [1991] OCCAM system has generalized knowledge about physical causality that is used to guide multistrategy learning. In contrast, we propose specific knowledge about classes of learning situations that can be used to guide learning strategy selection. Integration of these and other approaches is still an open research issue.

From the system design point of view, the use of Meta-XPs provides a number of benefits. Because Meta-XPs make the trace of reasoning explicit, an intelligent system can directly inspect the reasons supporting specific conclusions. This allows the system to build introspective explanations its own reasoning processes and determine what and how to learn in order to improve its own performance. The uniform architecture and the modularity of learning strategies makes it relatively straightforward to incorporate additional types of learning situations and additional learning strategies, and to extend the system to deal with different task domains involving different kinds of knowledge. Thus, this approach provides a robust framework for integrated learning approaches to real-world problems in which reasons for processing and learning decisions are made explicitly by the system rather than existing only in the minds of the system's designers.

5.3 Metacognition and learning

Much of the meta-knowledge research in artificial intelligence has focused on knowledge about knowledge, or knowledge about the facts that one does or does not know (see, for example, [Barr, 1979; Davis, 1979; Davis and Buchanan, 1977]). Much of the metacognition research in psychology has also focussed on similar issues, focussing on cognitive processes and knowledge having the self as referent. Of particular interest is the psychological research on meta-memory, which, in addition to knowledge about knowledge, includes knowledge about memory in general and about the peculiarities of one's own memory [Weinert, 1987]. The empirical results obtained from the Meta-TS system support the claim that meta-knowledge should also include knowledge about reasoning and learning strategies.

Experimental results in the metacognition literature suggests that introspective reasoning of the kind we propose here can facilitate reasoning and learning. For example, in a study by Alexander [1992], gifted children were found to have a better understanding of their own mental activities. These children, called "attributors," tended to form mental and perceptual explanations about their own behavior. In another study, Carr [1992] showed that strategic, deliberative control of mental processes facilitated the use of decomposition strategies for a mathematics problem solving task. The improved performance resulted from meta-knowledge about reasoning strategies and knowledge about when a strategy was appropriate to use. Our research extends these results by specifying computational mechanisms for metacognitive processing, focussing in particular on the selection and use of learning strategies.

There are four important ways that metacognitive knowledge and capabilities bear on work in introspective learning. First, and foremost, is the emphasis on cognitive self-monitoring. This behavior is a human's ability to read their own mental states during cognitive processing [Flavell and Wellman, 1977; Wellman, 1983; Wellman, 1985]. Thus, there is a moment-by-moment understanding of the content of one's own mind, and thus an internal feedback for the cognition being performed and a judgement of progress (or lack thereof). Psychological studies have confirmed a positive effect between meta-memory and memory performance in cognitive monitoring situations [Schneider, 1985; Wellman, 1983]. This directly supports the hypothesis that there must be a review phase or a parallel review process that introspects to some degree about the performance element in an intelligent system.

Second, our Meta-XP theory places a heavy emphasis on explicit representation. Trains of thought, as well as the products of thought, are represented as meta-knowledge structures, and computation is not simply calculated results from implicit side-effects of processing. This emphasis is echoed in Chi's [1987] argument that to understand knowledge organization and to examine research issues there must be some representational framework. Though diverging from the framework suggested by Chi, Meta-XP theory provides a robust form to represent knowledge about knowledge and process. For example, Meta-XPs can represent the difference between remembering and forgetting [Cox and Ram, 1992a]. Since forgetting is the absence of a successful outcome, this is difficult to capture in most frameworks. Forgetting is an important issue in machine learning [Markovitch and Scott, 1993] and in metamemory [Spear, 1978; Wellman and Johnson, 1979]. Meta-TS implements a simple form of forgetting in which obsolete knowledge is deleted once it is identified.

Third, because the approach taken by the introspective learning paradigm clearly addresses the issue of memory organization and the indexing problem, it can assign blame to errors that occur from mis-indexed knowledge structures and poorly organized memory. Although Meta-TS does not need to deal with the mis-indexed knowledge problem, extensions of this approach to other types of tasks and domains may need to do so, particularly if deep knowledge is required. Memory organization of suspended goals, background knowledge, and reasoning strategies is as important in determining the cause of a reasoning failure as are the goals, propositions and strategies themselves [Ram, Cox, and Narayanan , in press]. Thus, memory retrieval and storage issues are relevant in deciding what to learn and which learning strategy is appropriate. This claim is supported by the metamemory community's focus on organizational features of memory and their relation to the human ability to know what they know, even in the face of an unsuccessful memory retrieval. Extending both the Meta-TS and Meta-AQUA models to include a cognitive model of human memory (including memory organization) is an important issue for future research.

Finally, there is the ability of a person to assess the accuracy of a given response. In addition, because a person has a feeling of knowing, even when recall is blocked, a person can make efficient use of search. Thus search and elaboration is pursued when an item is on the "tip of the tongue" and abandoned when unfamiliarity is judged. This provides an efficiency heuristic to control memory search and avoid the combinatoric explosion of inferences [Lachman, Lachman, and Thronesbery, 1979]. To be able to model this dimension of human metaknowledge might require a two-layered memory architecture. In the lower level would be the actual memories, cases and propositions. The second layer would be a metamemory which stores memories of the first layer. This would account for the human ability to know what they know. This layer might be composed of Trace Meta-XPs, perhaps representing the memories of retrieving a past memory. Much work remains to be done to implement and test such a model, but our framework allows a natural integration of these ideas into the existing implementations.

One of the major differences between the manner in which humans learn and the manner in which machines do is that humans perform dynamic metacognitive monitoring or self-evaluation. Humans know when they are making progress in problem solving, even if they are far from a solution, and they know when they have sufficiently learned something with respect to some goal [Weinert, 1987]. They know how to allocate mental resources and can judge when learning is over. Many of the above reviews (e.g., [Chi, 1987; Schneider, 1985; Wellman, 1983]) cite evidence for such claims. Research in Meta-XP theory is a step in the direction in adding this metacognitive monitoring capability to AI systems, but this is well beyond the capabilities of Meta-TS at present.

It should be noted that the learning strategies represented in Meta-TS, or other strategy selection programs such as Meta-AQUA, are at a finer level of granularity than those examined by much of psychology. For example, it would be misleading to assert that the types of learning strategies studied by the metacognition community are similar to index learning, explanation-based generalization, and other learning strategies used in Meta-AQUA, although Meta-TS's strategies are closer in content to the cognitively plausible learning methods suggested by Anderson [1989] and others. Instead, metacognition research focusses on a person's choice of strategies at the level of elaboration or rehearsal. However, many of the results from this research do support the overall approach taken in this paper, that of using introspection to support the selection of appropriate strategies in different situations. Although we are currently building computer systems at what might be called the micro-level, it would be eventually be desirable to build systems that integrate the kinds of behavior exhibited by human learners at the macro-level as well.

Finally, we would like to emphasize that our model of learning is agnostic about the issue of "consciousness." Weinert [1987] argues convincingly that consciousness is a persistent unsolved problem in metacognition. However, we make no claims about when people are aware of their introspection, nor that active, strategic learning necessarily implies a conscious process. We would expect some of the processing in our model to be deliberative and conscious, especially when the reasoner becomes aware of a failure in its reasoning process, but it is evident that people possess and use metacognitive knowledge that they are sometimes not aware of. This issue is beyond the scope of and orthogonal to the point of this article.

6 Pragmatic implications of the model

While Meta-TS is intended as a model of learning, our results have several pragmatic implications for the design of intelligent tutoring systems. Major issues in developing an intelligent tutoring system include what to teach and how to teach; specific points of importance are the student model, the teacher model, the organization of knowledge, the simulation of the task, and the interface to the learner [Psotka, Massey, and Mutter, 1988; Spohrer and Kleiman, 1992]. Our research suggests that it would be valuable to teach shallow troubleshooting knowledge, including context-specific associative knowledge and general heuristic knowledge. Furthermore, since our model of learning involves reasoning about actual troubleshooting experiences, and active pursuit of identified learning goals through multiple learning strategies, we suggest that novice troubleshooters be placed in simulated or actual problem solving situations and encouraged to reason about what they are doing and why they are doing it. This approach is consistent with recent approaches suggested in the educational literature. For example, in Scardamalia and Bereiter's [1991] Teacher C model, the teacher is concerned with helping students formulate their own goals, do their own activation of prior knowledge, ask their own questions, direct their own inquiry, and do their own monitoring of comprehension. Redmond [1992] suggests a similar approach to learning through apprenticeship. His model is implemented in the CELIA system, which observes an expert troubleshooter (in this case, a car mechanic) solving the given problem, reasons explicitly about how it would solve the same problem, and determines what it needs to learn in order to be able to explain and predict the expert's behavior based on the differences between the expert's problem solving processes and its own.

Several researchers have proposed simulation environments in which students play roles that are connected to their goals, and whose successful completion requires acquisition of the skills to be taught (e.g., [Schank, Fano, Jona, and Bell , to appear; Shute, Glaser, and Raghavan, 1988; van Berkum, Hijne, de Jong, van Joolingen, and Njoo, 1991]). Van Berkum and his colleagues, for example, identify four aspects of the design of such systems: simulation models, learning goals, learning processes, and learning activity. In their model, students pursue learning goals with three dimensions: the type of knowledge, the representation of that knowledge, and the generality and applicability of that knowledge. Learning occurs through interaction with simulated environments using four types of learning actions (orientation, hypothesis generation, testing, and evaluation) which are guided by the learning goals. The learning model implemented in Meta-TS provides a basis for the design of such learning environments. In particular, we suggest that these environments provide facilities to encourage students to introspect, question, and explore. Exploring the relationship between learning and education is a fruitful direction for future research.

7 Conclusions

The focus of our research is on the integration of different kinds of knowledge and reasoning processes into goal-driven, real-world systems that can learn through experience. In particular, we are interested in modelling the kind of active, goal-driven learning processes that underlie deliberative learning during the performance of complex reasoning tasks. We have developed a computational model of introspective multistrategy learning, in which a reasoner introspects about its own performance on a reasoning task, identifies what it needs to learn to improve its performance, formulates learning goals to acquire the required knowledge, and pursues its learning goals using multiple learning strategies. Our approach relies on a declarative representation of meta-models for reasoning and learning. The resulting computational model represents a novel combination of metacognition and multistrategy learning and provides a framework for cognitive modelling as well as the design of artificial intelligence systems.

In this article, we have presented a particular case study of an introspective multistrategy learning system for the complex task of diagnostic problem solving on the assembly line of a real-world manufacturing plant. The research was based on observations of troubleshooting operators and protocol analysis of the data gathered in the test area of an operational electronics manufacturing plant. The model was implemented in a computer system, Meta-TS, which uses multiple types of knowledge to troubleshoot printed-circuit boards that fail in the test area. Meta-TS was evaluated on a series of troubleshooting problems, including actual problems encountered by the human operators in the manufacturing plant. The results were evaluated both qualitatively and quantitatively to determine the efficacy of the learning methods as well as to compare the model to human data. The results show that the model can be computationally justified as a uniform, extensible framework for multistrategy learning, and cognitively justified as a plausible model of human learning.

Acknowledgements

Research for this article was supported by the National Science Foundation under grant IRI-9009710 and by the Georgia Institute of Technology. We wish to thank Rick Alterman and Kurt VanLehn for their extremely helpful comments on an earlier draft of this article. The problem solving model presented in this paper benefited from discussions with Sally Cohen, Christine Mitchell, T. Govindaraj, Ashok Goel, and Leon McGinnis at Georgia Tech, and was based on data collected by Sally Cohen. We thank Jim Wallace, Don Haywood, Mike Bunger, and Kenneth Mitchell of the manufacturing plant at Atlanta for their cooperation during this study.

References

[Alexander, 1992] J. Alexander. Metacognition and Giftedness. Paper presented at the *SouthEast Cognitive Science Conference*, Atlanta, GA, January 1992.

[Anderson, 1989] J. Anderson. A Theory of the Origins of Human Knowledge. *Artificial Intelligence*, 30:313–351, 1989.

[Arabian, 1989] J. Arabian. *Computer Integrated Electronics Manufacturing and Testing*. Mercel Dekker, New York, 1989.

[Barr and Feigenbaum, 1981] A. Barr and E. A. Feigenbaum, editors. *The Handbook of Artificial Intelligence*, volume 1. Addison-Wesley, Reading, MA, 1981.

[Barr, 1979] A. Barr. Meta-Knowledge and Cognition. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 31–33, 1979.

[Bereiter and Miller, 1989] S. R. Bereiter and S. M. Miller. A Field-based Study of Troubleshooter's Information Utilization in Computer-Controlled Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):205–219, 1989.

[Birnbaum, Collins, Freed, and Krulwich, 1990] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-Based Diagnosis of Planning Failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 318–323, Boston, MA, 1990.

[Carbonell, 1986] J. G. Carbonell. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning II: An Artificial Intelligence Approach*. Morgan Kaufman Publishers, San Mateo, CA, 1986.

[Carr, 1992] M. Carr. Metacognitive Knowledge as a Predictor of Decomposition Strategy Use. Paper presented at the *SouthEast Cognitive Science Conference*, Atlanta, Georgia, January 1992.

[Chandrasekaran, 1988] B. Chandrasekaran. Generic Tasks as Building Blocks for Knowledge-based Systems: The Diagnosis and Routine Design Examples. *Knowledge Engineering Review*, 3:183–219, 1988.

[Chi and VanLehn, 1991] M. T. H. Chi and K. A. VanLehn. The Content of Physics Self-Explanations. *The Journal of the Learning Sciences*, 1(1):69–105, 1991.

[Chi, 1987] M. T. H. Chi. Representing Knowledge and Metaknowledge: Implications for Interpreting Metamemory Research. In F. E. Weinert and R. H. Kluwe, editors, *Metacognition, Motivation, and Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[Clancey, 1986] W. J. Clancey. *Knowledge-based Tutoring: The GUIDON Program*. MIT Press, Cambridge, MA, 1986.

[Cohen, Mitchell, and Govindaraj, 1992] S. M. Cohen, C. M. Mitchell, and T. Govindaraj. Analysis and Aiding the Human Operator in Electronics Assembly. In M. Helander and M. Nagamachi, editors, *Design for Manufacturability: A Systems Approach to Concurrent Engineering and Ergonomics*, pages 361–376. Taylor and Francis, London, 1992.

[Cohen, 1990] S. M. Cohen. A Model of Troubleshooting in Electronics Assembly Manufacturing. Master's thesis, Report # CHMSR 90-3, Georgia Institute of Technology, Center for Human-Machine Systems Research, Atlanta, GA, 1990.

[Cox and Ram, 1991] M. Cox and A. Ram. Using Introspective Reasoning to Select Learning Strategies. In R. S. Michalski and G. Tecuci, editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 217–230, Harpers Ferry, WV, November 1991.

[Cox and Ram, 1992a] M. T. Cox and A. Ram. An Explicit Representation of Forgetting. In *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics*, Baden-Baden, W. Germany, 1992.

[Cox and Ram, 1992b] M. T. Cox and A. Ram. Multistrategy Learning with Introspective Meta-Explanations. In *Machine Learning: Proceedings of the Ninth International Conference*, Aberdeen, Scotland, July 1992.

[Danyluk, 1991] A. P. Danyluk. Gemini: An Integration of Analytical and Empirical Learning. In R. S. Michalski and G. Tecuci, editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 191–206, Harpers Ferry, WV, November 1991.

[Davis and Buchanan, 1977] R. Davis and B. G. Buchanan. Meta-Level Knowledge: Overview and Applications. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977.

[Davis, 1979] R. Davis. Interactive Transfer of Expertise: Acquisition of New Inference Rules. *Artificial Intelligence*, 12:121–157, 1979.

[DeJong and Mooney, 1986] G. F. DeJong and R. J. Mooney. Explanation-Based Learning: An Alternative View. *Machine Learning*, 1(2):145–176, 1986.

[deKleer and Williams, 1987] J. deKleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.

[deKleer, Doyle, Steele, and Sussman, 1977] J. deKleer, J. Doyle, G. L. Steele, and G. J. Sussman. Explicit Control of Reasoning. *SIGPLAN Notices*, 12(8), 1977.

[Douglas, 1988] P. N. Douglas. Effective Functional Test Design Increases PCB Throughput. *Electronic Manufacturing*, pages 16–18, December 1988.

[Doyle, 1979] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[Falkenhainer, 1989] B. Falkenhainer. *Learning from Physical Analogies: A Study in Analogy and the Explanation Process*. Ph.D. thesis, University of Illinois, Department of Computer Science, Urbana, IL, 1989.

[Fink and Lusth, 1987] P. K. Fink and J. C. Lusth. Expert Systems and Diagnostic Expertise in the Mechanical and Electrical Domains. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17(3):340–347, May/June 1987.

[Fisher, 1987] D. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, 2:139–172, 1987.

[Flavell and Wellman, 1977] J. H. Flavell and H. M. Wellman. Metamemory. In J. R. V. Kail and J. W. Hagen, editors, *Perspectives on the Development of Memory and Cognition*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

[Gentner, 1989] D. Gentner. Mechanisms of Analogical Learning. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*. Cambridge University Press, London, 1989.

[Govindaraj and Su, 1988] T. Govindaraj and Y. D. Su. A Model of Fault Diagnosis Performance of Expert Marine Engineers. *International Journal of Man-Machine Studies*, 29:1–20, 1988.

[Hale, 1992] C. Hale. *Effects of Background Knowledge on Associative Learning in Causal Domains*. Ph.D. thesis, Georgia Institute of Technology, School of Psychology, Atlanta, GA, 1992.

[Hall, 1988] R. J. Hall. Learning by Failing to Explain: Using Partial Explanations to Learn in Incomplete or Intractable Domains. *Machine Learning*, 3:45–78, 1988.

[Hammond, 1989] K. J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Perspectives in Artificial Intelligence. Academic Press, Boston, MA, 1989.

[Hunter, 1990a] L. E. Hunter. Knowledge Acquisition Planning for Inference from Large Datasets. In B. D. Shriver, editor, *Proceedings of the Twenty Third Annual Hawaii International Conference on System Sciences*, pages 35–45, Kona, HI, 1990.

[Hunter, 1990b] L. E. Hunter. Planning to Learn. In *Proceedings of the Twelvth Annual Conference of the Cognitive Science Society*, pages 261–268, Boston, MA, July 1990.

[Kakani, 1987] T. Kakani. Testing of Surface Mount Technology Boards. In *Proceedings of the International Test Conference*, pages 614–620, Washington, DC, 1987.

[Katz and Anderson, 1987] I. Katz and J. R. Anderson. Debugging: An Analysis of Bug-location Strategies. *Human-Computer Interaction*, 3:351–399, 1987.

[Kuipers and Kassirer, 1984] B. Kuipers and J. Kassirer. Causal Reasoning in Medine: Analysis of a Protocol. *Cognitive Science*, 8:363–385, 1984.

[Lachman, Lachman, and Thronesbery, 1979] J. L. Lachman, R. Lachman, and C. Thronesbury. Metamemory Through the Adult Life Span. *Developmental Psychology*, 15(5):543–551, 1979.

[Laird, Rosenbloom, and Newell, 1986] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning*, 1:11–46, 1986.

[Lesgold, Lajoie, Bunzo, and Eggan, 1988] A. Lesgold, S. Lajoie, M. Bunzo, and G. Eggan. SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job. In J. Larkin, R. Chabay, and C. Scheftic, editors, *Computer Assisted Instruction and Intelligent Tutoring Systems: Establishing Communication and Collaboration*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Markovitch and Scott, 1993] S. Markovitch and P. D. Scott. Information Filtering: Selection Mechanisms in Learning Systems. *Machine Learning*, 10:113–151, 1993.

[Michalski and Stepp, 1983] R. S. Michalski and R. E. Stepp. Learning from Observation: Conceptual Clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufman, Los Altos, CA, 1983.

[Michalski and Tecuci, in press] R. S. Michalski and G. Tecuci, editors. *Machine Learning: A Multistrategy Approach, Volume IV*. Morgan Kaufman, San Mateo, CA, 1993. In press.

[Miller and Walker, 1988] R. K. Miller and T. C. Walker. *Artificial Intelligence Applications in the Computer/Electronics Industry*. SEAI Technical Publications/Fairmont Press, Madison, GA, 1988.

[Minton, 1988] S. Minton. *Learning Effective Search Control Knowledge: An Explanation-based Approach*. Ph.D. thesis, Technical Report CMU-CS-88-133, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, 1988.

[Mitchell, Keller, and Kedar-Cabelli, 1986] T. M. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1(1):47–80, 1986.

[Mooney and Ourston, 1991] R. J. Mooney and D. Ourston. A Multistrategy Approach to Theory Refinement. In R. S. Michalski and G. Tecuci, editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 115–130, Harpers Ferry, WV, November 1991.

[Morik, 1991] K. Morik. Balanced Cooperative Modeling. In R. S. Michalski and G. Tecuci, editors, *Proceedings of the First International Workshop on Multistrategy Learning*, pages 65–80, Harpers Ferry, WV, November 1991.

[Narayanan, Ram, Cohen, Mitchell, and Govindaraj, 1992] S. Narayanan, A. Ram, S. M. Cohen, C. M. Mitchell, and T. Govindaraj. Knowledge-Based Diagnostic Problem Solving and Learning in the Test Area of Electronics Assembly Manufacturing. In *Proceedings of the SPIE Conference on Applications of AI X: Knowledge-Based Systems*, Orlando, FL, 1992.

[Park and Wilkins, 1990] Y. Park and D. C. Wilkins. Establishing the Coherence of an Explanation to Improve Refinement of an Incomplete Knowledge Base. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 511–516, Boston, MA, 1990.

[Pazzani, 1991] M. J. Pazzani. Learning to Predict and Explain: An Integration of Similarity-Based, Theory-Driven and Explanation-Based Learning. *The Journal of the Learning Sciences*, 1(2):153–199, 1991.

[Pirolli and Bielaczyc, 1989] P. Pirolli and K. Bielaczyc. Empirical Analyses of Self-Explanation and Transfer in Learning to Program. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 450–457, Ann Arbor, MI, 1989.

[Pirolli and Recker, in press] P. Pirolli and M. Recker. Learning Strategies and Transfer in the Domain of Programming. *Cognition and Instruction*, in press.

[Pollock, 1989] J. L. Pollock. A General Theory of Rationality. *Journal of Theoretical and Experimental Artificial Intelligence*, 1:209–226, 1989.

[Psotka, Massey, and Mutter, 1988] J. Psotka, L. D. Massey, and S. A. Mutter, editors. *Intelligent Tutoring Systems: Lessons Learned.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Quilici, to appear] A. Quilici. Toward Automatic Acquisition of an Advisory System's Knowledge Base. *Applied Intelligence*, to appear.

[Quinlan, 1986] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[Rajamoney, 1989] S. Rajamoney. *Explanation-Based Theory Revision: An Approach to the Problems of Incomplete and Incorrect Theories*. Ph.D. thesis, University of Illinois, Department of Computer Science, Urbana, IL, 1989.

[Ram and Cox, in press] A. Ram and M. T. Cox. Introspective Reasoning using Meta-Explanations for Multistrategy Learning. In R. S. Michalski and G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV.* Morgan Kaufman Publishers, San Mateo, CA, 1993. In press. Also available as Technical Report GIT-CC-92/19, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1992.

[Ram and Hunter, 1992] A. Ram and L. Hunter. The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Applied Intelligence*, 2:47–73, 1992.

[Ram and Leake, in press] A. Ram and D. B. Leake. Learning, Goals, and Learning Goals. In A. Ram and D. B. Leake, editors, *Goal-Driven Learning*, chapter 1. MIT Press/Bradford Books, Cambridge, MA, in press.

[Ram, Cox, and Narayanan , in press] A. Ram, M. T. Cox, and S. Narayanan. Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In A. Ram and D. B. Leake, editors, *Goal-Driven Learning*, chapter 17. MIT Press/Bradford Books, Cambridge, MA, in press.

[Ram, 1991] A. Ram. A Theory of Questions and Question Asking. *The Journal of the Learning Sciences*, 1(3&4):273–318, 1991.

[Ram, 1993] A. Ram. Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. *Machine Learning*, 10:201–248, 1993.

[Rasmussen, 1984] J. Rasmussen. Strategies for State Identification and Diagnosis in Supervisory Control Tasks, and Design of Computer-Based Support Systems. *Advances in Man-Machine Systems Research*, 1:139–193, 1984.

[Redmond, 1992] M. Redmond. *Learning by Observing and Understanding Expert Problem Solving*. Ph.D. thesis, Georgia Institute of Technology, College of Computing, Atlanta, Georgia, 1992.

[Reich, in press] Y. Reich. Macro and Micro Perspectives of Multistrategy Learning. In R. S. Michalski and G. Tecuci, editors, *Machine Learning: A Multistrategy Approach, Volume IV.* Morgan Kaufman Publishers, San Mateo, CA, 1993. In press.

[Richardson, Keller, Maxion, and Polson, 1985] J. J. Richardson, R. A. Keller, R. A. Maxion, and P. G. Polson. Artificial Intelligence in Maintenance: Synthesis of Technical Issues. Technical Report F33615-82-C-0013, Air Force Human Resources Laboratory, October 1985.

[Riddle, 1992] P. Riddle, editor. *Proceedings of the Workshop on Integrated Learning in Real-World Domains, Ninth International Machine Learning Conference*, Aberdeen, Scotland, July 1992.

[Rouse and Hunt, 1984] W. Rouse and R. Hunt. Human Problem Solving in Fault Diagnosis Tasks. *Advances in Man-Machines Systems Research*, 1:195–222, 1984.

[Scardamalia and Bereiter, 1991] M. Scardamalia and C. Bereiter. Higher Levels of Agency for Children in Knowledge Building: A Challenge for the Design of New Knowledge Media. *The Journal of the Learning Sciences*, 1(1):37–68, 1991.

[Schank, Fano, Jona, and Bell , to appear] R. C. Schank, A. Fano, K. Jona, and B. Bell. The Design of Goal-Based Scenarios. *The Journal of the Learning Sciences*, to appear.

[Schank, 1983] R. C. Schank. The Current State of AI: One Man's Opinion. *The AI Magazine*, 4(1):3–8, 1983.

[Schneider, 1985] W. Schneider. Developmental Trends in the Metamemory-Memory Behavior Relationship: An Integrative Review. In D. L. Forrest-Pressley, G. E. MacKinnon, and T. G. Waller, editors, *Metacognition, Cognition, and Human Performance, Volume 1*. Academic Press, New York, NY, 1985.

[Shute, Glaser, and Raghavan, 1988] V. Shute, R. Glaser, and K. Raghavan. Inference and Discovery in an Exploratory Laboratory. Technical Report 10, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA, February 1988.

[Spear, 1978] N. E. Spear. *The Processing of Memories: Forgetting and Retention*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[Spohrer and Kleiman, 1992] J. C. Spohrer and R. Kleiman. Content = Knowledge + Media: Content Acquisition, Maintenance, Communication, and Construction of Intelligent Tutoring Systems. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pages 521–525, Chicago, IL, October 1992.

[Stefik, 1981] M. J. Stefik. Planning And Meta-Planning (Molgen: Part 2). *Artificial Intelligence*, 16:141–169, 1981.

[Stroulia and Goel, 1992] E. Stroulia and A. Goel. A Model-Based Approach to Incremental Self-Adaptation. In *ML-92 Workshop on Computational Architectures for Supporting Machine Learning and Knowledge Acquisition*, Aberdeen, Scotland, 1992.

[Sussman, 1975] G. J. Sussman. *A Computer Model Of Skill Acquisition*, volume 1 of *Artificial Intelligence Series*. American Elsevier, New York, 1975.

[Tecuci and Michalski, 1991] G. Tecuci and R. S. Michalski. A Method For Multistrategy Task-Adaptive Learning Based On Plausible Justifications. In *Machine Learning: Proceedings of the Eighth International Workshop*, Chicago, IL, June 1991.

[van Berkum, Hijne, de Jong, van Joolingen, and Njoo, 1991] J. J. A. van Berkum, H. Hijne, T. de Jong, W. R. van Joolingen, and M. Njoo. Aspects of Computer Simulations in an Instructional Context. *Education and Computing*, 6:231–239, 1991.

[VanLehn, Jones, and Chi, 1992] K. A. VanLehn, R. Jones, and M. T. H. Chi. A Model of the Self-Explanation Effect. *The Journal of the Learning Sciences*, 2(1):1–59, 1992.

[Veloso and Carbonell, 1991] M. Veloso and J. G. Carbonell. Automating Case Generation, Storage, and Retrieval in Prodigy. In R. S. Michalski and G. Tecuci, editors, *Proceedings Of The First International Workshop On Multistrategy Learning*, pages 363–377, Harpers Ferry, WV, November 1991.

[Weinert, 1987] F. E. Weinert. Introduction and Overview: Metacognition and Motivation as Determinants of Effective Learning and Understanding. In F. E. Weinert and R. H. Kluwe, editors, *Metacognition, Motivation, and Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[Wellman and Johnson, 1979] H. M. Wellman and C. N. Johnson. Understanding of Mental Process: A Developmental Study of "Remember" and "Forget". *Child Development*, 50:79–88, 1979.

[Wellman, 1983] H. M. Wellman. Metamemory Revisited. In M. T. H. Chi, editor, *Trends in Memory Development Research*. S. Karger AG, Basel, Switzerland, 1983.

[Wellman, 1985] H. M. Wellman. The Origins of Metacognition. In D. L. Forrest-Pressley, G. E. MacKinnon, and T. G. Waller, editors, *Metacognition, Cognition, and Human Performance, Volume 1*. Academic Press, New York, NY, 1985.

[Wilensky, 1984] R. Wilensky. Meta Planning: Representing and Using Knowledge about Planning in Problem Solving and Natural Language Understanding. *Cognitive Science*, 5:197–234, 1984.