January 1987

Hcport No. STAN-CS-87-1175
Also numbered KS L-86-62

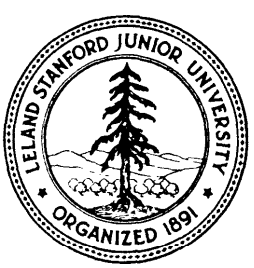
Using and Evaluating Differential Modeling in Intelligent Tutoring and Apprentice Learning Systems

by

D. C. Wilkins, W. J. Clancey, and B. G. Buchanan

Department of Computer Science

Stanford University
Stanford, CA 94305



Knowledge Systems Laboratory
KSL Report No. KSL-86-62

October 1986
Rev. 1, January 1987

Using and Evaluating Differential Modeling in Intelligent Tutoring and Apprentice Learning Systems

David C. Wilkins, William J. Clancey and Bruce G. Buchanan

Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

To appear in:

Intelligent Tutoring Systems: Lessons Learned, J. Psotka, D. Massey and S. Mutter, editors, Lawrence Erlbaum Publishers, 1987

Contents

Using and Evaluating Differential Modeling in Intelligent Tutoring and Apprentice Learning Systems

David C. Wilkins, William J. Clancey and Bruce G. Buchanan

Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

A powerful approach to debugging and refining the knowledge structures of a problem-solving agent is to differentially model the actions of the agent against a gold standard. This paper proposes a framework for exploring the inherent limitations of such an approach when a problem solver is differentially modeled against an expert system. A procedure is described for determining a *performance upper bound* for debugging via differential modeling, called the synthetic *agent method.* The synthetic agent method systematically explores the space of near miss training instances and expresses the limits of debugging in terms of the knowledge representation and control language constructs of the expert system.

1 Introduction

Artificial Intelligence has long been interested in methods to automatically refine and debug an intelligent agent. This is a central concern in machine learning and automatic programming, where the agent to be improved is a program. It is also a central concern in intelligent tutoring, where the agent to be improved is a human problem solver. Many AI systems for improving an intelligent agent involve

differential modeling of the agent against the observable problem-solving behavior of another agent. We focus on the situation where one of the agents is a knowledge-based expert system and the knowledge structures to be improved encode factual information that is declaratively represented[1].

This paper describes the synthetic agent method, which allows calculation of a *performance upper* bound on improvement to an intelligent agent attainable by differential modeling of the agent against an expert system. A performance upper bound identifies missing or erroneous knowledge in an intelligent agent that a particular differential modeling system is inherently incapable of identifying. By contrast, most performance evaluation procedures aim to determine a performance lower bound; they experimentally demonstrate that a particular differential modeling system can successfully identify some missing or erroneous knowledge.

The synthetic *agent method* involves replacing the human problem solver in a differential modeling scenario with a synthetic agent that is another expert system. The knowledge in the synthetic agent expert system is systematically modified to be slightly different than the knowledge in the original expert system. The knowledge in the synthetic agent is modified to be slightly 'better' in an apprenticeship learning scenario and slightly 'worse' in an intelligent tutoring scenario.

This paper is organized as follows. Section 2 surveys previous and current work on improving an intelligent agent via differential modeling. Section 3 identifies important performance evaluation issues related to evaluation of a differential modeler. Section 4 presents and discusses the synthetic agent method. Finally, Section 5 describes an application of the synthetic agent method that is currently underway.

This paper presents our framework for evaluating a differential modeling system. No experimental results are given. A future paper will describe the use of the framework to evaluate the ODYSSEUS modeling program (described in Section 5) in the context of intelligent tutoring and apprenticeship learning.

[1] As much domain-specific knowledge as possible is declaratively represented in a well designed knowledge-intensive expert system. Domain-specific procedural knowledge is contained in an expert system shell for the generic problem class (Clancey, 1984).

2 The Process of Differential Modeling

Many AI systems that debug and refine an intelligent agent employ a method called *differential modeling;* this is the process of identifying differences between the observed behavior of a problem-solving agent and the behavior that would be expected in accordance with an *explicit model* of problem solving.

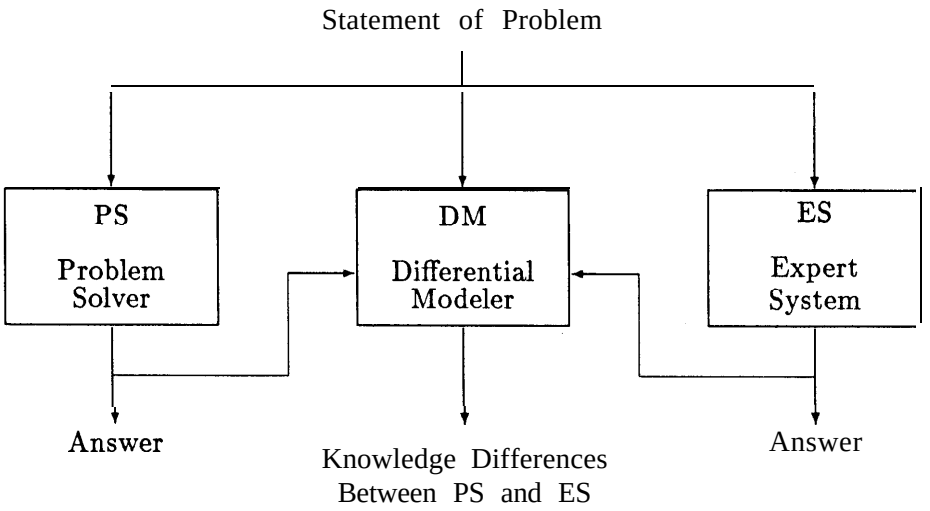


Figure 1: A general model of the differential modeling process. PS solves a problem, and DM finds differences between knowledge structures of PS and ES. In this paper, equal attention is given to the situation of apprenticeship learning where PS is a human expert and the goal is to improve ES; and the situation of intelligent tutoring where PS is a student and the goal is to improve PS.

The differential modeling process is illustrated in Figure 1. The three major elements are a problem solver (PS), a differential modeler (DM), and a knowledge-based expert system (ES). The task of the DM is to identify differences between the knowledge structures of PS and ES in the course of watching PS solve a problem, for example a medical diagnosis problem. In the figure, Answer consists of all observable behavior of the respective problem solver. The DM can be quite complex and can easily exceed the complexity of the ES.

Two major tasks that confront a DM are global and local credit assignment, which are performed by a global and local learning critic, respectively. The global critic determines when the observable behavior of PS suggests a difference between the knowledge structures of ES and PS. In such a situation, the local critic is summoned to identify possible knowledge differences between ES and PS that are suggested by the actions of PS. A complete learning system consists of a global critic, local critic and a repair component (Dietterich and Buchanan, 1981); discussion of the repair stage is beyond the scope of this paper.

2.1 Previous work in differential modeling

AI systems that employ a differential modeling approach to debugging and refining a problem-solving agent are found in the areas of machine learning, automatic programming, and intelligent tutoring. We first describe systems that do not employ a knowledge-based expert system as the explicit model of problem solving and then describe systems that do.

The earliest such systems were in the area of machine learning, notably, Samuel's checker player and Waterman's poker player (Samuel, 1963; Waterman, 1970). The PS used by Samuel's DM program was a book of championship checker games. The DM global critic task was accomplished by comparing the move of PS to the move that Samuel's program made in the same situation. The local critic task was accomplished by adjusting the coefficients of a polynomial evaluation function for selecting moves so that the action of the program equaled the action of PS. A recent example of machine learning research that uses a differential modeling approach is the PRE system for theory-directed data interpretation (Dietterich, 1984). PRE learns programs for Unix commands from examples of the use of the commands. The DM employs constraint propagation to identify differences between the PS and the programs for commands.

In automatic programming, the synthesis of LISP and PROLOG functions from example traces falls under the rubric of debugging via differential modeling (Biermann, 1978; Shapiro, 1983). The PS consists of the input/output behavior of a

correct program. The DM modifies the program being synthesized whenever it does not give the same output as PS when given the same input.

In intelligent tutoring the goal is to 'debug' a human problem solver. Many intelligent tutoring systems contain an expert system and use a differential modeling technique, including the WEST program in the domain of games (Burton and Brown, 1982), SOPHIE III and GUIDON in the domain of diagnosis (Brown et al., 1982; Glancey, 1979), and the MACSYMA-ADVISOR in the domain of symbolic integration (Genesereth, 1982). SOPHIE III uses an expert system for circuit diagnosis as an aid in isolating hypothesis errors in the behavior of students who are performing electronic troubleshooting. GUIDON is built over the MYCIN expert system for medical diagnosis (Buchanan and Shortliffe, 1984); student hypothesis errors are discovered in the process of conducting a Socratic dialogue.

Recent research within machine learning also uses an expert system as the explicit model of problem solving, especially within the subarea of *apprenticeship* learning. Apprenticeship learning is defined as a form of learning that occurs in the context of normal problem solving and uses *underlying theories* of the problem solving domain to accomplish learning. Examples of apprenticeship learning systems are LEAP and ODYSSEUS. The LEAP program refines knowledge bases for the VEXED expert system for VLSI circuit design (Mitchell et al., 1985). PS is a circuit designer who is using the VEXED circuit design aid and the underlying theory used by the DM is circuit theory. ODYSSEUS refines and debugs knowledge bases for the HERACLES expert system shell, which solves problems using the heuristic classification method (Wilkins, 1986). When the ODYSSEUS problem domain is medical diagnosis, PS is a physician diagnosing a patient. The DM uses two underlying theories, the principal one being a strategy theory of the problem-solving method. ODYSSEUS is also applicable to intelligent tutoring; it functions as a student modeling program for the GUIDON2 intelligent tutoring system (Glancey, 1986a).

2.2 Assumptions and issues in differential modeling

Much of the power of an expert system derives from the quantity and quality of its domain-specific knowledge. For the purposes of this paper, the principal function of the differential modeler is to find factual domain knowledge differences between the problem solver and the expert system. Our work assumes that the expert system represents domain knowledge declaratively, including domain-specific control knowledge. Further, as much as possible, the knowledge is represented independently of how it will be used by problem-solving programs. This practice facilitates use of the same domain knowledge for different purposes, such as problem solving, explanation, tutoring and learning.

The framework provided by this paper for understanding the limits of debugging via differential modeling has been fashioned with the following assumptions in mind. First, we assume that an agent is differentially modeled against a knowledge-based expert system that is capable of solving the problems presented to the human PS. Second, we assume that the observed actions of the agent consist of normal problem-solving behavior in a domain. And third, we assume that the goal of the differential modeling system is to discover factual domain knowledge differences between the agent and the expert system's knowledge base, as opposed to the discovery of procedural control knowledge differences; procedural knowledge involves sequencing constructs such as looping and recursion.

There are many open questions regarding debugging via differential modeling against an expert system. For instance, what are the types of knowledge in the PS that can and cannot be debugged using a differential modeling approach? What characteristics and organization of an ES facilitate differential modeling? What characteristics and organization impose inherent limitations? How can the strengths and weaknesses of a particular DM be best described? The evaluation methodology proposed in this paper, called the synthetic agent method, provides a framework for the exploration of these questions.

3 Performance Evaluation Issues

DM performance evaluation is intimately related to ES performance evaluation. The function of a DM is to improve the performance of an ES and so DM performance evaluation requires ES performance evaluation. Although ES evaluation is a difficult and time consuming task, there is agreement on the general approach that should be taken when ES is an expert system program. Examples of performance evaluation studies based on a sound methodology are the evaluations of the MYCIN, INTERNIST and RL expert systems (Yu et al., 1979; Miller et al., 1984; Fu and Buchanan, 1985).

Two major functions of a DM are global and local credit assignment[2]. The general problem of assessing the limits of a DM consists of finding performance upper bounds on a DM's global and local critics. The difficulty of these functions is very domain dependent. In the domain used to develop repair theory, the global critic merely has to determine whether a student's answer to a subtraction problem is correct (Brown and VanLehn, 1980). Sometimes a DM has a person perform the-global credit assignment, for example in LEAP and MACSYMA-ADVISOR. In very difficult domains a DM might have a person perform both global and local credit assignment; TEIRESIAS takes this approach when debugging MYCIN (Davis, 1982). TEIRESIAS can be viewed as an intelligent editor that allows an expert to perform global and local credit assignment while watching MYCIN solve problems.

In domains where expertise involves heuristic problem solving, having a program perform global credit assignment is often very difficult. In a medical apprenticeship, a student may recognize that his or her knowledge is deficient when he or she can no longer make sense of the sequence of questions that the physician asks the patient. Since a weakly plausible explanation for any sequence of questions often exists, this can be very difficult to implement in a computer program. A similar situation exists in complex games such as chess or checkers. There is usually no way to know that a given move is necessarily bad; it depends on what follows. Samuel's checker player solved the global critic problem by declaring a discrepancy to exist

[2]Recall from section 2 that the global critic notices that something is wrong and the local critic determines which part of the knowledge base is responsible for the error. A learning program consists of a global and local critic and a repair component.

whenever the expert (e.g., the book move) and the checker program recommended different moves at a particular board configuration.

There are often many different changes the local critic can make to effect an improvement in the performance element. The selection process is usually based on. which modification leads to the best improvement in the performance element. Selection is very much affected by how 'improvement' in defined. This is further discussed in Section 3.2.

3.1 Performance evaluation and the synthetic agent method

The synthetic agent method proposed in this paper is considerably different from standard performance evaluation methods in two fundamental ways. The purpose of the remainder of Section 3 is to explain and justify these aspects of the synthetic agent method. In Section 3.2 we argue that a fruitful evaluation criteria for a knowledge-based system should be quality of the individual knowledge elements, not the quality of the problem solving performance of a particular problem-solving program. These metrics only partially overlap and certainly conflict in the short term. In section 3.3, we describe how the focus of the proposed synthetic agent method is to delineate *a performance upper bound.* A performance upper bound describes where and under what conditions a debugging system for a problem solver must fail. By contrast, a standard evaluation approach aims at showing the extent to which a debugging system can succeed. Further, instead of characterizing the limits of debugging in terms of a percentage of problems that cannot be solved, the synthetic agent method characterizes the performance upper bound in terms of the knowledge representation language and the inference constructs used in the expert system.

3.2 Knowledge-oriented vs. performance-oriented validation

The ultimate goal of a DM is to improve the performance of a PS or ES. The architecture of knowledge-based systems requires a shift in our concept of improved performance. We refer to the type of validation technique we advocate as *knowledge-oriented* validation and distinguish it from the traditional practice of *performance-oriented* validation.

Performance-oriented validation requires that modifications to a particular problem-solving program improve problem-solving performance. Because this type of validation has traditionally focused on improved performance with respect to a single problem-solving program, the veracity of the underlying knowledge has not been of overriding concern. A system designed exclusively to maximize problem-solving performance of a particular problem-solving program may use a method of knowledge representation in which the semantics of the domain knowledge cannot be represented easily, if at all. A polynomial evaluation function for rating checker positions, for example, captures none of the meaning of its terms.

Knowledge-oriented validation might be defined as performance-oriented validation that prohibits lessening the truth of individual knowledge elements solely for the sake of problem-solving performance. The advent of large declarative knowledge bases used by multiple problem solving programs makes this perspective important. Examples of multiple problem-solving programs that might use the same medical knowledge base are programs to accomplish medical diagnosis, knowledge acquisition, intelligent tutoring, and explanation. When multiple programs use the same declaratively-specified factual knowledge base, it is helpful to specify knowledge in a manner that is independent, so far as possible, of its use. Knowledge-based validation accomplishes this by requiring that changes to the knowledge base be semantically meaningful.

Suppose we wish to be faithful to the traditional performance-oriented validation paradigm when using multiple-purpose knowledge bases. This requires that every time a learning program finds a change to the knowledge base that will improve

one problem-solving program, before that change can be recorded, the validation method must insure that the aggregate performance of all programs is improved. This policy will be expensive and computationally overwhelming. Further, programs for all the intended uses of a knowledge base are not necessarily in existence at the time learning is taking place.

Another rationale for knowledge-oriented validation is our belief that performance in the long term will be more correct and robust if the knowledge structures are carefully developed. Moreover, when PS is a person, it is unrealistic, probably even unwise, to attempt to replace semantically-rich knowledge structures with others that deviate radically from them merely to improve short-term performance.

It should be noted that to some extent all programs for improving an intelligent agent aim at both good performance and good knowledge; nevertheless, almost all past research in machine learning, intelligent tutoring and automatic programming has adopted a pure performance-oriented validation approach. This is especially true in automatic programming, where any mutation to the program to be debugged is judged to be acceptable if it causes the program to produce the correct output when given a correct input/output training instance (Shapiro, 1983).

In machine learning, one of the best systems for refining an expert system knowledge base is the SEEK2 program for the EXPERT expert system shell (Ginsberg et al., 1985). This learning system takes a performance-oriented validation approach. One possible input to SEEK2 is a *representative set* of past solved cases and an initial knowledge base of rules. Given this input, SEEK2 attempts to modify elements of the knowledge base so as to maximize the problem-solving performance of the EXPERT expert system on the given representative set of solved problem cases. In EXPERT, the strengths of inexact rules in the knowledge base are represented using certainty factors (CFs). Examples of modification operators used by SEEK2 to improve performance are LOWER-CF and RAISE-CF (Ginsberg, 1986). When a representative set of past cases is present, the strengths of inexact rules are determined, since certainty factors can be given a strict probabilistic interpretation (Heckerman, 1986). We strongly believe that an arbitrary change to the strength of a rule just to improve performance is unjustifiable and unnecessary (Wilkins and Buchanan,

1986). The cost of this improved performance is a knowledge base that may contain incorrect knowledge. The SEEK2 refinement approach is not an instance of knowledge-oriented learning; it does not use knowledge-oriented validation.

A good example of knowledge-oriented learning is repair theory in the domain of subtraction problems (Brown and VanLehn, 1980). Repair theory is concerned with detecting underlying bugs, given the observable problem solving behavior of students. Repair theory has a procedural model of problem solving that claims to be a plausible model of the associated human skill; bugs of students are correlated with possible bugs in the problem-solving procedure for subtraction. Repair theory is similar in spirit to the synthetic agent method we propose for assessing a differential modeling system. Repair theory generates most of the significant possible bugs by deleting parts of the procedural knowledge; likewise we expect our approach to generate most of the significant possible types of bugs in the declarative domain knowledge base, mainly by deleting parts of the knowledge base, as we shall describe in Section 4. The main difference is that in the repair theory model of subtraction the PS and ES knowledge is almost completely procedural, whereas we are interested in factual knowledge is declaratively represented.

3.3 Capability-oriented vs. limitation-oriented validation

A typical way of validating that a DM improves an ES involves using a disjoint set of validation and training problem sets. The ES solves the *validation* problem set and its performance is recorded. Then the DM improves the ES while watching a human expert PS solve a *training* problem set. Finally ES solves the validation problems again; the amount of improvement in performance provides a measure of the quality of the DM.

This scenario establishes a lower bound on the quality of a DM. By increasing the size of the training problem set, DM might improve ES even more. We refer to validation methods that establish a lower bound on the quality of a DM as *capability-oriented.* For a given set of training and validation problems, capability-oriented validation shows that the DM is responsible for a more capable ES.

Another method of validating a DM is to have the DM watch a student solve a training problem set. Let us assume that the student exhibits a representative set of the types of domain knowledge errors that could be made in the problem domain. A domain expert can manually identify the domain knowledge errors connected with each training problem. This manual analysis provides a performance upper bound with respect to this training set for the DM, and the DM modeling program is measured against this standard. The goals of this type of manual analysis and our proposed automated analysis using the synthetic agent method are identical, in the case where the student and the problem set have been both constructed so as to allow all possible types of domain knowledge errors to be made.

We desire to know those types of differences in an expert system knowledge base that cannot be detected or corrected via differential modeling. In contrast to the capability-oriented approach, our validation approach aims at determining when the differential modeler must fail — *we* are *limitation-oriented.* For example, a limit of a program for inducing LISP functions from examples might be that the program can't induce cases that require certain types of loop constructs. In our work, we have focused on showing certain conditions that force the differential modeling approach to fail under the most favorable of conditions, the single fault assumption. The multiple fault assumption would allow determination of a broader performance upper bound.

4 Synthetic Agent Method of Validation

The apprenticeship learning and tutoring scenarios shown in Figures 2 and 3 involve two agents: a person and an expert system. The person serves as an expert and student, in the context of apprenticeship learning and intelligent tutoring, respectively. The *synthetic agent method* consists of replacing the person with a synthetic agent, which is another expert system, in order to experiment with and validate the differential modeling system objectively. The knowledge in the synthetic agent expert system is modified to be slightly different from the knowledge in the original expert system. The knowledge is modified to be slightly 'better' in an apprenticeship

learning scenario and slightly 'worse' in an intelligent tutoring scenario.

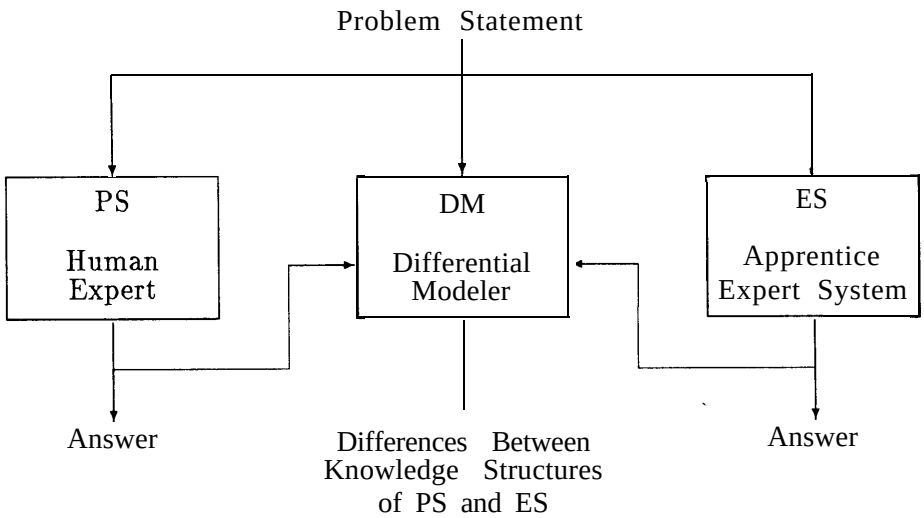


Figure 3: Apprentice learning scenario: Apprentice expert system watches human expert through the differential modeling program, with the goal of improving the apprentice program's knowledge.

An advantage of the synthetic agent method is control over interpersonal variables involved in differential modeling. An example of an interpersonal variable is the problem-solving style of a PS, as exemplified by the set of strategic diagnostic operators used by the PS. Diagnostic operators specify the permissible task procedures that can be applied to a problem as well as the allowable methods for achieving the task procedures. Examples of problem-solving operators in the domain of diagnosis include: ask general questions, ask clarifying questions, refine hypotheses, differentiate between hypotheses, and test hypothesis. Another interpersonal variable is the quantity of domain-specific knowledge that the PS possesses.

While control of interpersonal variables almost always leads to an incorrect DM performance lower bound, conclusions reached concerning a performance upper bound are sound when interpersonal variables are controlled. If a system is inherently limited under the most optimal assumptions possible for differential modeling, it will still be inherently limited in those settings that involve a less optimal differential modeling setting.

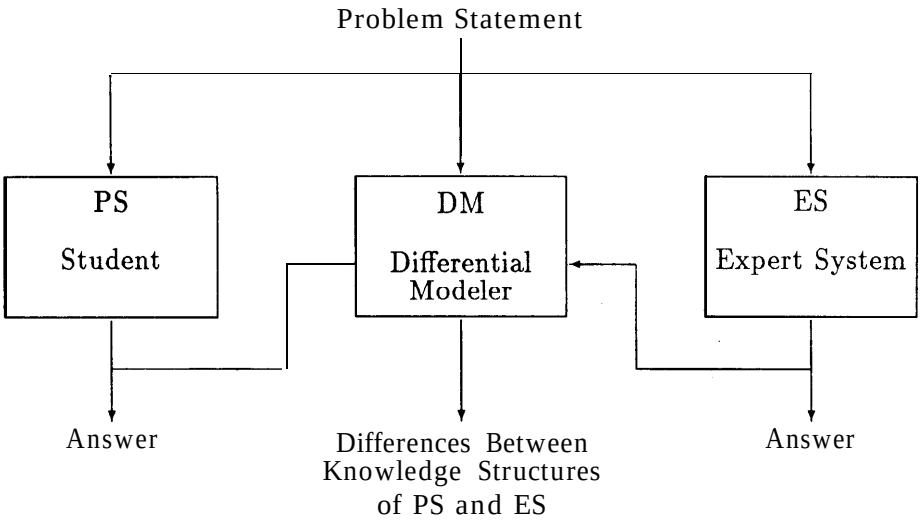


Figure 4: Intelligent tutoring scenario: Expert system watches student through the differential modeling program, with the goal of improving the student's knowledge.

In the learning and tutoring scenarios, the synthetic agent method treats the original expert system knowledge base as a "gold standard". The apprentice ES and the student PS always have a deficiency with respect to this gold standard. In this paper we restrict our analysis to the situation where the apprentice's knowledge differs from the gold standard by a single element of knowledge; hence we have a single fault assumption. Two types of knowledge base discrepancies are possible: missing knowledge and erroneous knowledge. The synthetic agent method procedure described in section 4.1 shows how deletion of knowledge can represent the space of missing and erroneous knowledge. Other methods for creating erroneous knowledge are described in section 4.3.

For a given problem statement, a distinction is made between referenced, observable, and essential knowledge in the ES's knowledge base. The relation between these categories is illustrated in Figure 4. Referenced knowledge is simply knowledge that is accessed during a problem solving case. *Observable knowledge* is knowledge whose removal leads to different external observable behavior of a PS, either in the sequence of actions that the PS exhibits or the final answer. *Essential knowledge* is

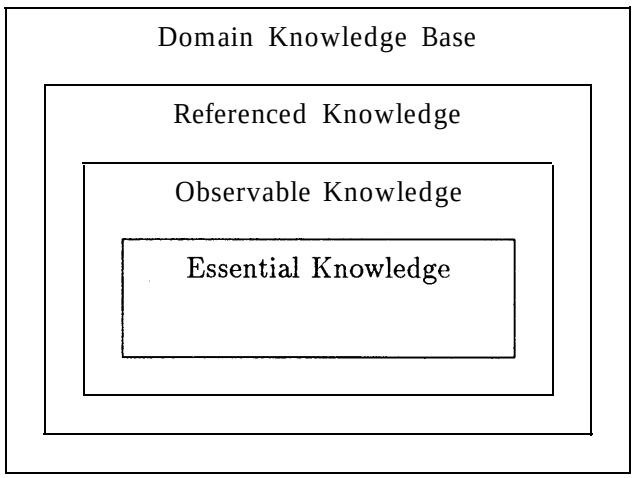


Figure 5: The relation between different categories of knowledge, with respect to a particular problem case.

knowledge whose removal leads to a significantly different final answer.

Of most concern is the apprentice's ability to acquire the *essential knowledge elements* connected with a problem statement. These are the relations most important for solving a given case. For plausible reasoning systems, what comprises a significantly different answer needs to be specified. For instance, if there are multiple diagnoses, the significance of the order in which the hypotheses are ranked needs to be determined. Acquisition of elements that are observable but not essential are also of interest, since they can be essential elements with respect to another problem statement.

The procedure for calculating a performance upper bound on a differential modeling system is now presented.

4.1 The synthetic agent method

Step 1. Create synthetic agent. Replace PS with a synthetic agent: a copy of ES with initially the same domain knowledge.

Step 2. Solve problem case. Solve a problem using PS and save the *solution trace,* i.e., the observable actions of PS and the final answer.

Step 3. Identify observable knowledge. For a particular problem case, collect all elements in the knowledge base that were referenced by PS during problem solving. Identify the *observable* knowledge: the subset of the referenced knowledge whose removal would lead to a different solution trace or a different final answer.

Step 4. For each observable knowledge element:

Step 4a. Remove the element from ES. In an apprenticeship learning scenario this creates an apprentice expert ES with missing knowledge. In an intelligent tutoring scenario the element removed from the ES is declared to be erroneous[3]. Since the element is still present in PS, the synthetic student PS has erroneous knowledge.

Step 4b. Detect and localize knowledge discrepancy. Have the PS solve the problem case. See if DM can detect (the global critic problem) and localize (the local critic problem) the knowledge difference.

Step 5. For each observable knowledge element:

Step 5a. Remove the element from PS. In an intelligent tutoring scenario this creates a synthetic student PS with *missing* knowledge. In an apprenticeship learning scenario the element removed from the PS is declared to be erroneous[3]. Since the element is still present in ES, the apprentice expert ES has erroneous knowledge.

Step 5b. Detect and localize knowledge discrepancy. Have the PS solve the problem case. See if DM can detect (the global critic problem) and localize (the local critic problem) the knowledge difference.

[3] N.B. This element of knowledge is treated as erroneous for purposes of validation. In reality, the element is true knowledge.

4.2 Discussion of synthetic agent method

An expert system's explanation facility can be helpful in locating the observable knowledge with respect to a given problem case. One of the hallmarks of a good expert system is its ability to explain its own reasoning. So it is not too much to ask for those pieces of knowledge used on a problem case, and a good explanation system might even be able to identify the essential knowledge. At worst, given the pieces of knowledge that were used to solve a particular problem, the essential pieces of knowledge can be determined by experimentation. Usually, only a small amount of an expert system's domain knowledge is observable with respect to a given problem, and our experiences in the medical diagnosis domain have shown us that only a small amount of the observable knowledge is essential knowledge.

Some knowledge that is referenced by the expert system may not have observable consequences, even if it is used by the problem solver, since the removal of knowledge does not always effect the external behavior of a problem solver. For instance, in MYCIN and NEOMYCIN, terms that represent medical symptoms and measurements, such as patient weight, have an ASKFIRST property. The expert system uses the value of this property to decide whether the value of a variable is first determined by asking the user or first determined by derivation by some other method, such as from first principles. However, if the system does not possess techniques for deriving the information from other principles, then the external behavior of the system is the same regardless of the value of the ASKFIRST property.

When testing the global critic in steps 4b and 5b of the synthetic agent method, part of the assessment must relate to whether the apprentice detects knowledge base differences close to the point in the problem-solving session where the different knowledge was used. This temporal proximity is important, since the problem-solving context at this point in the problem-solving session strongly focuses the search for missing or erroneous knowledge.

4.3 Categories of errors

The knowledge organization that we focus upon specifies all factual domain knowledge in a declarative fashion. In such a knowledge base, there are two main categories of errors: missing and erroneous knowledge. Missing knowledge is absent from the knowledge base, and erroneous knowledge is factually incorrect knowledge that is present in the knowledge base.

The space of missing knowledge is easy to generate, especially with the single fault assumption. Recall that the original expert system serves as our gold standard and the domain knowledge in the expert system is declaratively represented. Hence, the number of single faults from missing knowledge is equal to the number of elements in the declarative knowledge base.

The space of erroneous knowledge is much more difficult to describe. The synthetic agent method takes a novel approach to the problem in steps 5a and 6a. An erroneous element is created by declaring a correct knowledge element to be erroneous for purposes of validation. We are also considering other approaches. Much of the knowledge is represented declaratively and typed. Therefore, erroneous knowledge can be generated by substituting different values for the knowledge in the range of the type, as long as the assumption can be made that the erroneous knowledge is at least correctly typed by the problem solver. The space of possible variations of declarative associational rule knowledge is significantly reduced by the practice used in the HERACLES' expert system shell of factoring different types of knowledge from the domain knowledge, such as causal, definitional, and control knowledge (Clancey, 1986b).

5 Application of Synthetic Expert Method

Our investigations of a performance upper bound for a differential modeler are being performed in the context of the HERACLES and ODYSSEUS systems. HERACLES is an expert system shell that solves classification-type problems using the heuris-

tic classification method (Clancey, 1985). The ODYSSEUS program differentially models a PS against any ES implemented using the HERACLES expert system shell (Wilkins, 1986). When PS is a human expert, ODYSSEUS functions as a knowledge acquisition program for the HERACLES expert system shell. When PS is a student, ODYSSEUS functions as a student modeling program for the GUIDON2 intelligent tutoring system, which is built over HERACLES.

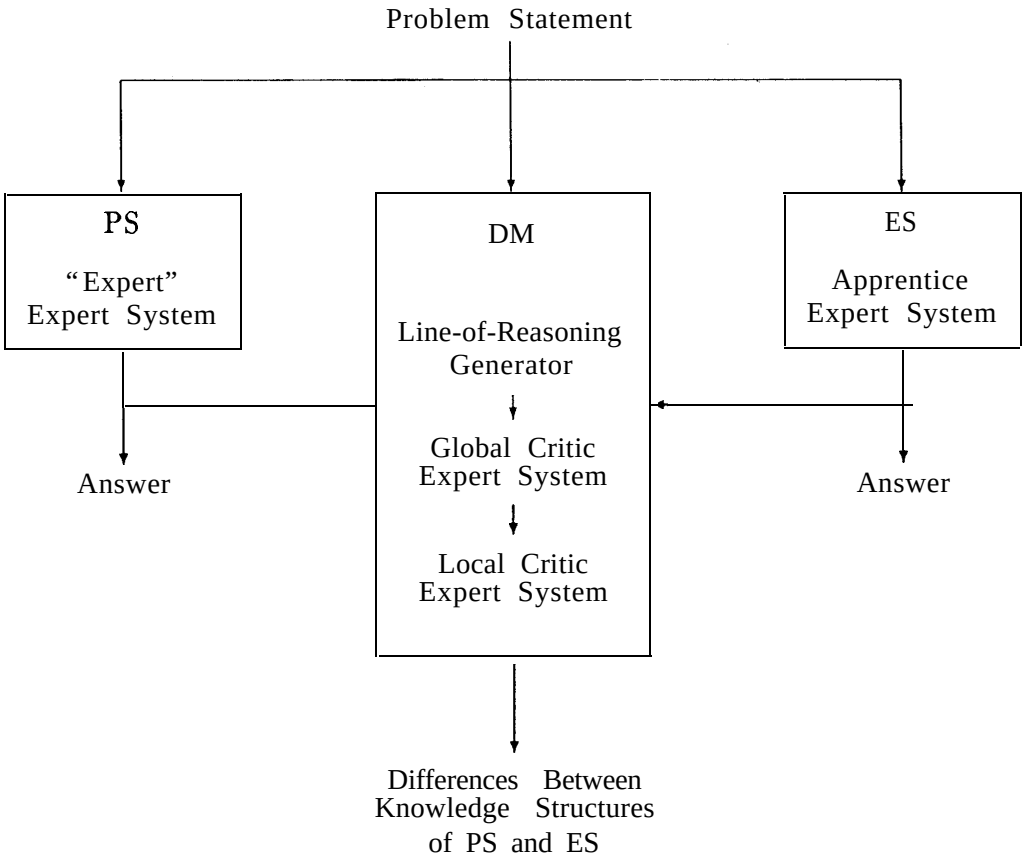


Figure 6: Synthetic agent validation situation for apprenticeship learning in which the role of the PS has been filled by a synthetic expert system. In apprenticeship learning, the DM watches PS to improve ES's knowledge structures.

In HERACLES, domain knowledge is encoded using a relational language and MYCIN-type rules (Clancey, 198613). The knowledge relations of the relational language are predicate calculus representations of the domain knowledge, written using

the logic programming language MRS. For example, an instantiation of the proposition (SUGGESTS $PARM $HYP) represents the fact that if a particular parameter is true then this suggests that a particular hypothesis is true. An instantiation of the template (ASKFIRST $FINDING $FLAG-VALUE) specifies whether the system should first ask the user for the value of a finding, or derive the information from existing information. The major domain knowledge base for HERACLES at this time is the NEOMYCIN knowledge base for diagnosing meningitis and neurological problems (Clancey, 1984). A second effort in the sand casting domain is called CASTER (Thompson and Clancey, 1986).

Three aspects of the HERACLES expert system shell facilitate the task of differential modeling faced by ODYSSEUS. First, distinctions are made between the different types of knowledge in HERACLES' knowledge base, such as heuristic, definitional, causal, and control knowledge. Second, the method of reasoning, called hypothesis-directed reasoning, approximates that used by human experts (Clancey, 1984). Hence, HERACLES can be viewed as a simulation of an expert's process of diagnosis. Third, the control knowledge is explicitly represented as a procedural network of subroutines and metarules that are both free of domain knowledge; the subroutines and metarules use variables rather than specific domain terms (Clancey, 1986b). By contrast, the heuristic rules in MYCIN have a great deal of control knowledge imbedded in the premises of the rules (Clancey, 1983; Buchanan and Shortliffe, 1984).

Figure 5 shows the place of the ODYSSEUS DM in the context of debugging an apprentice expert system. The DM tracks the problem-solving actions of the PS step by step. For each observable step of the problem solver, ODYSSEUS generates and scores the alternative lines of reasoning that can explain the reasoning step. If the global critic does not find any plausible reasoning path, or all found paths have a low plausibility, ODYSSEUS assumes that there is a difference in knowledge between the human problem solver and the expert system. The local critic attempts to locate the knowledge difference either automatically or by asking the expert specific questions. ODYSSEUS) analysis of problem-solving steps uses two underlying domain theories: a strategy theory of the problem-solving method called hypothesis-directed reasoning using the heuristic classification method, and an inductive predictive theory for

heuristic rules that uses a library of previously solved problem cases.

The ODYSSEUS global and local critics are themselves being implemented as two HERACLES-based expert systems. There are three reasons why we choose to implement the critics as expert systems. First, the task that confronts the learning critics is a knowledge-intensive task (Dietterich and Buchanan, 1981), and expert system techniques are useful for representing large amounts of knowledge. Second, with an expert system architecture, the reasoning method used by the critics can be made explicit and easily evaluated, since the domain knowledge is declaratively encoded using HERACLES' knowledge relations and simple heuristic rules. Third, since ODYSSEUS is designed to improve any HERACLES-based expert system, it can theoretically improve itself in an apprenticeship learning setting.

Approximately sixty different knowledge relations in HERACLES specify the declarative domain knowledge. It would be useful to know how successful the global and local critics are at detecting discrepancies in the different knowledge relations of the knowledge representation language. Are there certain types of knowledge relations whose absence is always noticeable? Are there particular types of knowledge whose absence is very hard to recognize? For example, HERACLES represents final diagnoses in a hierarchical tree structure; determining that a problem is caused by a missing link in this structure may be very difficult for the apprentice to discover. By contrast, it may be very easy to discover whether a trigger property of a rule is missing. A trigger property causes the conclusion of a rule to treated as an active hypothesis if particular clauses of the rule premise are satisfied. Clearly global and local credit assignment are greatly affected by the complexity of the procedural control knowledge used in the expert system shell.

6 Summary

With the proliferation of expert systems, methods of intelligent tutoring and apprenticeship learning that are based on differential modeling of the normal problem-solving behavior of a student or expert against a knowledge-intensive expert system

should become increasingly common. The synthetic agent method is proposed as an objective means of assessing the limits of a particular differential modeling program in the context of intelligent tutoring and apprenticeship learning. The power of a differential modeler is crucially dependent upon the expert system's method of knowledge representation and control. The synthetic agent method provides a means of expressing the limitations of a differential modeler in terms of the knowledge representation and control vocabulary.

The synthetic agent method involves a systematic perturbation of a program that takes the place of the student or expert. Traditionally, methods of evaluating a differential modeler have focused on a performance lower bound. The described synthetic agent method focuses on establishing a performance upper bound. It provides a means of exploring the extent that a differential modeling system is able to detect and isolate an arbitrary difference between a knowledge base of an expert system and the problem-solving knowledge of a student or expert. Our work to date confirms our belief that the task of differential modeling is easier the more an expert system represents factual domain knowledge in a declarative fashion.

The validation framework described in this paper is being used to assess the limits of the ODYSSEUS modeling program in the context of intelligent tutoring and apprenticeship learning. Students and experts are being differentially modeled against knowledge bases for the HERACLES' expert system shell. This should lead to a better understanding of the synthetic agent method, the ODYSSEUS modeling program, and the extent to which HERACLES' method of knowledge representation and control facilitates differential modeling.

7 Acknowledgments

We are grateful for very substantive critiques of draft versions of this paper by Marianne Winslett, Haym Hirsh and Devika Subramanian.

This work was supported in part by NSF grant MCS-83-12148, ONR/ARI contract N00014-79C-0302, Advanced Research Project Agency (Contract DARPA N00039-83-C-0136), the National Institute of Health (Grant NIH RR-00785-11),

National Aeronautics and Space Administration (Grant NAG-5-261), and Boeing (Grant W266875). We are grateful for the computer time provided by SUMEX-AIM and the Intelligent Systems Lab of Xerox PARC.

8 References

Biermann, A. W. (1978). The inference of regular LISP programs from examples. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-8(8):585-600.

Brown, J. S., Burton, R., and DeKleer, J. (1982). Pedagogical and knowledge engineering techniques in SOPHIE I, II and III. In Sleeman, D. H. and Brown, J. S., editors, *Intelligent Tutoring Systems,* London: Academic Press.

Brown, J. S. and VanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills. *Cognitive Science,* 4:479–426.

Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Mass.: Addison-Wesley.

Burton, R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In Sleeman, D. H. and Brown, J. S., editors, *Intelligent Tutoring Systems,* London: Academic Press.

Clancey, W. J. (1979). *Transfer of Rule-Based Expertise Through a Tutorial Dialogue.* PhD thesis, Stanford University. Stanford Technical Report STAN-CS-79-769.

Clancey, W. J. (1983). The epistemology of a rule-based system: a framework for explanation. *Artificial Intelligence,* 20:215–251.

Clancey, W. J. (1984). NEOMYCIN: reconfiguring a rule-based system with application to teaching. In Clancey, W. J. and Shortliffe, E. H., editors, *Readings in Medical Artificial Intelligence,* chapter 15, pages 361–381, Reading, Mass.: Addison-Wesley.

Clancey, W. J. (1985). Heuristic classification. *Artificial Intelligence,* 27:289–350.

Clancey, W. J. (1986a). From GUIDON to NEOMYCIN to HERACLES in twenty short lessons. *AI Magazine,* 7:40–60.

Clancey, W. J. (1986b). Representing control knowledge as abstract tasks and metarules. In Coombs, M. and Bolc, L., editors, *Computer Expert Systems,* Springer Verlag. Also, Knowledge Systems Lab Report KSL-85-16, Stanford University, April 1985.

Davis, R. (1982). Application of meta level knowledge in the construction, maintenance and use of large knowledge bases. In Davis, R. and Lenat, D. B., editors, *Knowledge-Based Systems in Artificial Intelligence,* New York: McGraw-Hill.

Dietterich, T. G. (1984). *Constraint Propagation Techniques for Theory-Driven Data Interpretations.* PhD thesis, Stanford University. Stanford Technical Report STAN-CS-79-xXx.

Dietterich, T. G. and Buchanan, B. *G.* (1981). *The Role of the Critic in Learning Systems.* Report HPP-81-19, Stanford University. Reprinted in Selfridge, O., Rissland, E. and Arbib, M. eds: Adaptive control of Ill-defined systems.

Fu, L. and Buchanan, B. *G.* (1985). *Inductive knowledge acquisition for rule based expert systems.* Technical Report KSL 85-42, Stanford University, Computer Science Dept.

Geneseret h, M. R. (1982). Diagnosis using hierarchical design models. In *Proceedings **of** the Second National Conference on Artificial Intelligence,* pages 278-283, American Association for Artificial Intelligence.

Ginsberg, A. (1986). A metalinguistic approach to the construction of knowledge base refinement systems. In *Proceedings **of** the 1986 National Conference on Artificial Intelligence,* pages 436-441.

Ginsberg, A., Weiss, S., and Politakis, P. (1985). SEEK2: a generalized approach to automatic knowledge base refinement. In *Proceedings **of** the 1985 IJCAI,* pages 367-374.

Heckerman, D. (1986). Probabilistic interpretations for Mycin's certainty factors. In Kanal, L. and Lemmar, J., editors, *Uncertainty in Artificial Intelligence,* North Holland.

Miller, R. A., Pople, H. E., and Myers, J. D. (1984). INTERNIST-l: an experimental computer-based diagnostic consultant for general internal medicine. In Clancey, W. J. and Shortliffe, E. H., editors, *Readings in Medical Artificial Intelligence,* chapter 8, pages 190-209, Reading, Mass.: Addison-Wesley.

Mitchell, T. M., Mahadevan, S., and Steinberg, L. I. (1985). LEAP: a learning apprentice for VLSI design. In *Proceedings **of** the 1985 IJCAI,* pages 573-580.

Samuel, A. L. (1963). Some studies in machine learning using the game of checkers. In Feigenbaum, E. and Feldman, D., editors, *New York: Computers and Thought,* McGraw-Hill.

Shapiro, E. H. (1983). *Algorithmic Program Understanding.* Cambridge: MIT Press.

Thompson, T. and Clancey, W. J. (1986). A qualitative modeling shell for process diagnosis. *IEEE Software,* 3(2):6–15.

Waterman, D. (1970). Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence,* 1:121–170.

Wilkins, D. C. (1986). Knowledge base debugging using apprenticeship learning techniques. In *Knowledge Acquisition for Knowledge-Based Systems Workshop,* pages 40.0-40.14.

Wilkins, D. C. and Buchanan, B. G. (1986). On debugging rule sets when reasoning under uncertainty. In *Proceedings of the 1986 National Conference on Artificial Intelligence,* pages 448-454.

Yu, V. L., Fagan, L. M., Wraith, S. M., and Clancey, W. J. (1979). Evaluating the performance of a computer-based consultant. *J. Amer. Med. Assoc.,* 242(12):1279–1282.