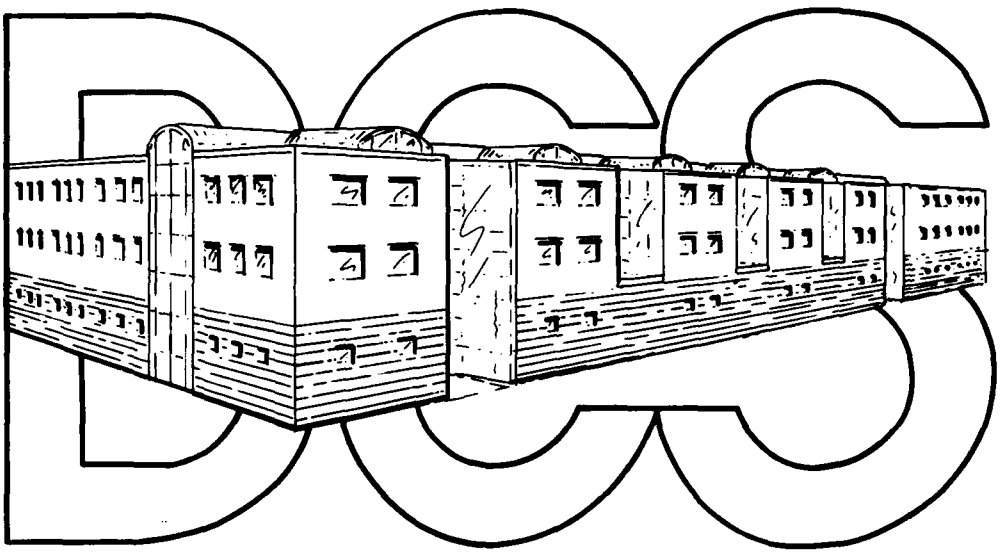
DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



THE NEW ADDITION

REPORT NO. UIUCDCS-R-89-1536

UILU-ENG-89-1755

An Analysis of the Distinction Between Deep and Shallow Expert Systems

by

Peter D. Karp and David C. Wilkins

August 1989

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UIUCDCS-R-89-1536 UILU-ENG-89-1755	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION University of Illinois 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Artificial Intelligence (Code 1133) Cognitive Science (Code 1142CS)
6c. ADDRESS (City, State, and ZIP Code) Dept. of Computer Science 1304 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) Office of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-88K-0124
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO
61153N	RR04206	0C	443g-008

11. TITLE (Include Security Classification)

An Analysis of the Distinction Between Deep and Shallow Expert Systems

12. PERSONAL AUTHOR(S)

Peter D. Karp and David C. Wilkins

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical Report	FROM 87-12-1 TO 91-1-30	1989, August	43

16. SUPPLEMENTARY NOTATION

Available as Report UIUCDCS-R-89-1536, Dept. of Computer Science, University of Illinois, Urbana, IL 61801.
To Appear in: **International Journal of Expert Systems.**

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
05	09	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
expert systems, knowledge representation, first-principles knowledge, causal knowledge, empirical knowledge, compilation, multi-level domain models, multi-purpose domain models, performance degradation, brittleness, and novel problem solving.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The first generation of expert systems (e.g., MYCIN, DENDRAL, R1) is often characterized as only using *shallow* methods of representation and inference, such as the use of production rules to encode empirical knowledge. First-generation expert systems are often dismissed on the grounds that shallow methods have inherent and fatal shortcomings which prevent them from achieving problem-solving behaviors that expert systems should possess. Examples of such desirable behaviors include graceful performance degradation, the handling of novel problems, and the ability of the expert system to detect its problem-solving limits.

This paper analyzes the relationship between the techniques used to build expert systems and the behaviors they exhibit to show that there is not sufficient evidence to link the behavioral shortcomings of first-generation expert systems to the shallow methods of representation and inference they employ. There is only evidence that the shortcomings are a consequence of a general lack of knowledge. Moreover, the paper shows that the first-generation of expert systems employ both shallow methods and most of the so-called deep methods. Lastly, we show that deeper methods augment but do not replace shallow reasoning methods; both are necessary.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Dr. Alan Meyrowitz, Dr. Susan Chipman	(202) 696-4302, 696-4320	1133 and 1142CS

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Department of Computer Science
Report No. UIUCDCS-R-89-1536

College of Engineering
UILU-ENG-89-1755
August 1989

An Analysis of the Distinction Between Deep and Shallow Expert Systems

Peter D. Karp
Knowledge Systems Lab
Department of Computer Science
Stanford University
Stanford, CA 94305

David C. Wilkins
Knowledge Based Systems Group
Department of Computer Science
University of Illinois
Urbana, IL 61801

To appear in:
International Journal of Expert Systems, 1989

Contents

An Analysis of the Distinction Between Deep and Shallow Expert Systems

Peter D. Karp
Knowledge Systems Lab
Department of Computer Science
Stanford University
Stanford, CA 94305

David C. Wilkins
Knowledge Based Systems Group
Department of Computer Science
University of Illinois
Urbana, IL 61801

Abstract

The first generation of expert systems (e.g., MYCIN, DENDRAL, R1) is often characterized as only using *shallow* methods of representation and inference, such as the use of production rules to encode empirical knowledge. First-generation expert systems are often dismissed on the grounds that shallow methods have inherent and fatal shortcomings which prevent them from achieving problem-solving behaviors that expert systems should possess. Examples of such desirable behaviors include graceful performance degradation, the handling of novel problems, and the ability of the expert system to detect its problem-solving limits.

This paper analyzes the relationship between the techniques used to build expert systems and the behaviors they exhibit to show that there is not sufficient evidence to link the behavioral shortcomings of first-generation expert systems to the shallow methods of representation and inference they employ. There is only evidence that the shortcomings are a consequence of a general lack of knowledge. Moreover, the paper shows that the first-generation of expert systems employ both shallow methods and most of the so-called deep methods. Lastly, we show that deeper methods augment but do not replace shallow reasoning methods; most expert systems should possess both.

Keywords

expert systems, deep expert systems, shallow expert systems, empirical knowledge, structure-function knowledge, first-principles knowledge, causal knowledge, production rules, multi-purpose problem solving, multi-level domain models, brittleness, declarative knowledge representation, performance degradation, novel problem solving, compilation.

1 Introduction

The distinction between deep and shallow expert systems arose in the early 1980s from a simple intuition: that the expert systems which had been built up until that time had serious limitations. Previous writings on this topic discussed these limitations in terms of behaviors that expert systems were unable to achieve, e.g., (Davis, 1984; Genesereth, 1984; Hart, 1982; Forbus, 1988). These analyses then took the logical step of examining the techniques that were used to construct these systems under the assumption that the behavioral limitations of these programs could be traced to these techniques. New techniques and research goals were proposed that were directed towards achieving the desired behaviors.

This paper shows that there are three problems with previous analyses. First, some behaviors are so poorly defined that it is impossible to determine whether a given program exhibits them or not. Second, some arguments as to how the potential behaviors of these programs are limited by the techniques used to build them are not convincing. And lastly, the new techniques proposed to overcome these limitations often bear a striking resemblance to existing techniques that were used in the first generation of expert systems.

A major thesis of this paper is that the deep/shallow distinction has its origins in, and can be distilled down to, one central hypothesis: that to attain more sophisticated behaviors, the next generation of "deep" expert systems will need to bring much more knowledge to bear on the problem-solving task than the first generation of "shallow" expert systems. One might view this claim as trivial since the conjecture that "in the knowledge lies the power" has existed for many years (Feigenbaum, 1977). Unfortunately, this conjecture makes the job of constructing more powerful programs sound too easy since it says that to do so we merely give them more knowledge. It does not tell us what knowledge to include, how to structure it, nor how to reason with it to solve different problems efficiently.

The remainder of this paper is structured as follows. Section 2 outlines behaviors that knowledge based systems should exhibit, such as graceful degradation near the limits of problem solving and the ability to solve novel problems. Section 3 analyzes techniques that appear relevant to obtaining these behaviors, such as use of first-principles knowledge and knowledge compilation.

2 Behavioral Goals for Expert Systems

Expert systems are programs that solve problems using a large amount of domain-specific knowledge, usually in a domain requiring human expertise, such as medical diagnosis. The external observable characteristics that expert systems should possess can be described in terms of behavioral goals. In this section, behavioral goals are defined, and the successes and failures of first-generation expert systems to achieve these goals are described.

The determination that a given behavioral goal *cannot* be attained using existing techniques is difficult. Theoretical approaches are too weak to offer definitive answers, so there is a need to rely on experimental evidence. Strictly speaking, conclusive experimental evidence can only be of a positive sort. An empirical approach can only definitively show that it is *possible* to build a given type of program, by presenting an example of such a program. But such an approach cannot show that is is *impossible* to build a given type of program. Negative evidence resulting from the failure to build a certain sort of program can never be definitive since it can always be argued that the experimenters lacked the skill, commitment, or resources to achieve the behavioral goal. In practice, of course, as such negative evidence accumulates it is assigned more and more credibility. But negative examples are extremely rare in the field of expert systems at present. There must be a general skepticism regarding claims about the limitations of existing techniques, given these methodological considerations.

2.1 Problem-Solving Explanations

A distinguishing feature of some of the earliest expert systems, such as MYCIN, was the ability to provide 'what' and 'how' explanations of problem-solving behavior (Buchanan and Shortliffe, 1984; Clancey, 1983). Such a capability is an immediate consequence of the representation of expertise using production rules, and the use of production rules as the major method of inference. An explanation of a problem-solving action is obtained by unwinding the instantiated production rules associated with a problem-solving action. The NEOMYCIN program builds upon the MYCIN framework and encodes most of the problem-solving strategy for the heuristic classification

problem-solving method in meta-level production rules (Clancey, 1984). This enables NEOMYCIN to provide explanations of problem-solving strategy as well as explanations of domain-level problem solving actions.

The behavioral goal of providing straightforward problem-solving explanations has been demonstrated by existing expert systems, such as MYCIN. Indeed, the ability to provide explanations is viewed as one of the defining traits of an expert system. While this behavioral goal has been largely met, future research in this area faces two major challenges.

The first major challenge relates to tailoring explanations to individual users, as is routinely done by human experts. To achieve this, more sophisticated explanation programs for expert systems will need to incorporate a user model that can tailor an explanation to the user. For example, a different explanation of the mechanism behind a liver disorder should be given to a high school student, a medical student, and a physician. More sophisticated explanation programs will also be able to give the same explanation at different levels of detail.

The use of more complex and deep methods of problem solving will provide a challenge for explanation programs. Inference will occur in a variety of ways, and hence the inferences connected with a given problem-solving action will not be limited to chaining of production rules. More information will be involved in any one problem-solving action, and hence there will be a need to abstract the essentials of an explanation from the myriad problem-solving details. The ABEL program is a medical diagnosis program that provides causal explanations at various levels of detail; each level gives a coherent account of the patient's case (Patil et al., 1981).

2.2 Graceful Performance Degradation

Experts usually have a narrow range of expertise. Yet, an expert's problem-solving abilities will usually degrade gracefully near the limits of the expertise, such as on peripheral problems. Also, an expert is usually aware when a problem is near or beyond his or her expertise – this is termed *limit detection.* Expert systems should exhibit such behavior. It is claimed that expert systems cannot degrade gracefully, that they are brittle (Lenat and et al, 1986; Holland, 1986).

A major reason for the brittleness of expert systems is simply that they are computer programs, and programs are brittle. Programs generally can't handle cases for which they have not been programmed. This problem is usually alleviated in conventional computer programming by devoting the majority of the programming effort and the majority of the code to handling exceptions and non-standard input. Because expert systems have been principally constructed in research labs, this common practice has not been followed. Another common method used in conventional programs is to have an error handling routine. An error check is generated if the program encounters something unexpected, such as divide by 0 or a type check, and control is passed to an error handling routine. Considering that the most basic programming practices for making programs less brittle have not been followed, it is not surprising that, in practice, expert systems exhibit brittleness.

There has been almost no research on understanding the phenomena of brittleness in expert systems, such as measuring how rapidly a system's performance degrades as problems become more and more peripheral, nor research to determine how extensive a task it would be to encode the knowledge required for peripheral problems. There have not been attempts to apply relatively straightforward solution approaches to this problem, such as those methods described above that are used in the construction of conventional computer programs. The only major effort is the CYC project for encoding a large amount of common sense knowledge (Lenat and et al, 1986). But this is a long-term effort and the way an actual expert system would use such knowledge is beyond the scope of the CYC project. There have not been any studies in understanding the difficulty in adding new knowledge when an expert system's performance exhibits brittleness on a peripheral problem. Indeed, it would seem that performance degradation would provide a powerful basis for automated knowledge acquisition.

While it is generally not recognized, some of the earliest expert systems possessed some form of limit detection. For example, the MYCIN program informs its user if it believes a patient's problem is outside its scope of expertise. It is not clear how difficult it is to build expert systems that have sophisticated knowledge of their own limitations; we know of no negative evidence in the literature of attempts to build such systems. It might be possible to make limit detection very sophisticated by constructing a companion expert system whose purpose is to evaluate whether or not a given problem is within another expert system's range of expertise; no one has

attempted this.

Thus, this behavioral requirement is an important but rather nebulous one about which AI has little hard evidence. It is difficult to evaluate the degree to which existing systems suffer from these limitations nor the effort required to satisfy this requirement.

2.3 Problem-Solving Speed

Most problem-solving is resource-limited: a solution must be obtained within a given period of time or using less than a certain amount of computational resources. Human expertise is nicely tailored to satisfy resource limitations. For example, based on very sparse information, a physician has to search a very large space of potential l diagnoses in a very short amount of time. A physician is able to achieve this by storing a very large amount of knowledge and by the use of a highly-compiled form of diagnostic expertise.

Most expert systems are highly knowledge-intensive and perform only a limited amount of inference. Yet, most expert systems research from MYCIN onwards has been conducted at the limits of the available computational speed and memory resources. Programs such as the TEIRESIAS program for interactive knowledge acquisition had to be run in separate stages because of memory limitations (Davis, 1982). Despite impressive strides in AI hardware over the last decade, large expert-system building tools such as KEE and Knowledge Craft are often still too slow on large problems to be very effectively used. This situation may be brought under control over the next few years by the introduction of more powerful AI workstations.

Currently, the use of more complete reasoning techniques, such as reasoning with a complete schematic of a circuit to diagnosis the circuit, is almost always several orders of magnitude slower than diagnosis using compiled expertise within the same problem domain. This is especially true for large problems.

2.4 Novel Problem Solving

It has been argued that it is desirable for an expert system to be able to solve "novel" problems, and that existing expert systems are unable to do so. The notion of novelty is another difficult behavioral requirement to define precisely. Davis (Davis, 1984) appears to view novel problems as those which the designer of the expert system did not anticipate while building the system. For example, since a programmer must anticipate every disease MYCIN diagnoses by writing one or more explicit production rules, MYCIN is not capable of diagnosing a novel disease. The method by which MYCIN accomplishes diagnosis may however appear novel to a physician that is observing the expert system's behavior.

The ability to solve novel problems is sufficiently imprecise that depending upon one's interpretation it is either impossible to achieve, or is routinely achieved by existing programs. It can be understood more clearly by considering the distinction between expert systems that use generative candidate descriptions, and those that use enumerative candidate descriptions. This distinction refers to whether an expert system derives the cases it reasons about from a pre-enumerated set, or whether it generates these cases dynamically.

For example, consider diagnostic problem-solving. One way to view the task that a diagnostic system confronts is as follows. Given information about the expected structure of a device, plus a description of its actual behavior, the task is to find the actual (malfunctioning) structure of the device. Usually the actual structure includes a malfunctioning component, but it could include extra components, such as solder bridges in circuits. A general approach to this problem is to consider a set of candidate device structures – the case descriptions – and determine which candidate's predicted behavior matches the observed behavior of the actual device most closely (Reiter, 1987; DeKleer and Williams, 1987).

These candidates can have two possible origins. Candidate structures and their associated behaviors can be retrieved from pre-enumerated classes that the program has stored, or they can be generated by the program. Usually the generative approach breaks the description of a complex device into many components, uses a set of pre-defined operators to introduce defects into selected components, and then computes the behavior of the new aggregate device. The enumerative approach stores all

different possible malfunctioning structures and their associated behaviors.

MYCIN uses the enumerative approach. It has a list of possible disease states (structures) and their associated symptoms (behaviors), and its rules match these behaviors against cases it is presented with to determine a diagnosis. DART (Genesereth, 1984) and Davis' system (Davis, 1984) use the second approach. They start with one model of the structure and behavior of a computer system and generate a potentially large set of other device models from this prototype, which are matched against the case at hand. The generation process is guided by the structure of the device. DART does the generation by trying to prove that each component of a device is broken, e.g., that a component it has been told is an AND gate is in fact not behaving like an AND gate. Davis' system uses a similar technique called *constraint suspension* to generate candidate devices. Candidates are generated by alternately suspending application of the constraints that describe the behaviors of different components of a device, and then simulating the outputs of that component by copying them from the empirically observed outputs of the component. Thus the behavior of the device model is coerced to match the behavior of the actual device.

The decision as to whether to employ generative or enumerative descriptions in a given expert system is based on two considerations. First is the classic trade-off between storage space and computation time. For many problems it is not feasible to store all the relevant candidates explicitly, such as all possible malfunctioning structures of a computer system, even under the single fault assumption. Second, a generation algorithm must exist. For some problems there may be no generation algorithm which is sufficiently fast and sufficiently constrained. For example, it is not possible to generate all the ways in which a human body and all known infectious bacteria could interact to produce observable symptoms. The theory of how this interaction occurs is incomplete: biological science cannot provide us with the operators needed to construct the space of "candidate device descriptions". Thus, MYCIN contains a list of those disease states that medical science has encountered.

Thus far we have distinguished the notions of stored versus generated candidates. It is also instructive to blur this distinction to consider Davis' hypothesis that "reasoning from first principles offers the possibility of dealing with novel faults. As we have seen, our system does not depend for its performance on a catalog of observed error manifestations" (Davis, 1984). Davis is essentially contrasting generative with

enumerative systems. This hypothesis is hard to evaluate because the notion of novelty is a hard one to pin down in AI programs. Davis seems to define novelty in terms of programmer forethought, i.e., novel situations are those which the programmer has not considered beforehand. Thus Davis claims that in the enumerative framework the author of a program must carefully consider every possible case the program will encounter and encode the solution for that case. If the program encounters a "novel" case that its author did not think of, the program will likely fail. Under the generative framework, Davis claims that the program will be able to generate any conceivable fault, and thus cases that would be novel to the program's author will not be novel to the program.

This novelty hypothesis is not convincing. Under *both* frameworks the author of the program must think about what *classes of cases* the program will encounter. When Davis' circuit diagnosis program was constructed its authors must have considered what classes of faults are usually observed in digital circuits so they would know what classes of fault-generation operators to catalog within the program. And under both frameworks cases may be described sufficiently generally that they match unanticipated problems and produce either correct or incorrect solutions. MYCIN's enumerative framework clearly does not require the author to think about every distinct patient whose symptoms will be described to MYCIN, but only about classes of patients. Both methods allow the program to consider clusters of problem cases which the programmer must define ahead of time. And the programmer must consider these cases if the program is to solve them correctly for reasons other than luck.

The decision to use generative versus enumerative descriptions is an engineering decision that depends upon such issues as the existence of a tractable generation algorithm, the computational complexity of this algorithm, and the size of the case space. There will be times when it is easier for the programmer to list a set of generation operators and rules for combining them, and times when it will be easier to enumerate the classes of cases. The generative approach is well suited to the domain of the systems of Davis and Genesereth for circuit diagnosis; the generation algorithm is simple and complete, although very computationally expensive. The enumerative approach is well suited to MYCIN's domain because no generation algorithm exists.

2.5 Multi-Purpose Problem Solving

The possession of expertise allows an expert to do much more than just get the right answer to a problem. In addition, experts often have the capability to explain the reasons for their problem solving actions, teach domain knowledge and problem solving skills to a novice or another expert, explain the observed problem solving behavior of a novice or another expert, and realize when their problem solving knowledge is inadequate to solve a particular problem. Lastly, when at an impasse, an expert can often elicit from another expert the exact knowledge necessary to solve the problem. Expert systems should be able to exhibit such multi-purpose problem solving in their domain of expertise. It would be especially convenient if one knowledge base for a domain could support these diverse dimensions of expertise. This is an important design goal for an expert system.

The achievement of multi-purpose problem solving appears to be highly dependent on making the knowledge in the expert system as modular, explicit, and declarative as possible (Buchanan and Shortliffe, 1984). Consequently, developing methods of knowledge representation that emphasize these characteristics has been a driving force in expert systems research from its earliest days. The method of knowledge representation used by MYCIN was certainly a large step in the direction of explicitly specifying domain knowledge in a modular fashion. Knowledge was chunked into production rules that were individually comprehensible to domain experts. This representation was able to support problem solving and the generation of "how" and "why" explanations. The TEIRESIAS (Davis, 1982) and GUIDON (Clancey, 1979) showed that MYCIN's representation provided the basis for programs to accomplish to varying degrees, interactive knowledge acquisition and intelligent tutoring, respectively.

The NEOMYCIN program was a reconstruction of MYCIN that made the method of knowledge representation and inference even more declarative and explicit, especially with respect to knowledge of strategy. This allowed the NEOMYCIN program to provide strategic and as domain-level explanations, and further facilitated the use of the same knowledge for multiple purposes. For example, the GUIDON-WATCH and GUIDON-MANAGE programs provide ways of teaching the knowledge in a NEOMYCIN knowledge base to a student (Clancey, 1986). And the ODYSSEUS program shows how the method of knowledge representation of NEOMYCIN provides the basis for explaining the observed actions of a student or an expert, and also the basis of expanding the

domain-level knowledge via apprenticeship learning (Wilkins, 1988b; Wilkins, 1988a).

2.6 Problem-Solving Sophistication

There is a belief that "deep systems will solve problems of significantly greater complexity than surface systems can" (Chandrasekaran and Mittal, 1983). One might support this intuition of Chandrasekaran by noting that most expert systems constructed to date have contained at most several thousand rules and are capable of solving at most a tiny subset of the problems in a domain such as medicine.

However, as we noted at the beginning of Section 2, such a lack of positive evidence hardly constitutes convincing negative evidence. One could just as easily construe existing systems to be proof that these techniques are valid for a sizable subset of a domain such as internal medicine (Pople, 1982), and that it is only a matter of time before knowledge bases are constructed using existing techniques that span all of medicine.

AI has no accepted way of measuring the complexity of a *problem domain* that would enable one to describe the types of problems that existing techniques are able to solve, and the classes of unsolvable problems. One way to measure the complexity of a problem domain is by measuring the complexity of a program that covers the problem domain. However we lack a principled method for this. Even if we had a principal method it would only provide an upper bound and this assumes that it can be proven that the program is correct. Possible measurements suggested by Buchanan for measuring the complexity of an expert system are knowledge base size, solution space size, and average inference complexity for different expert systems (Buchanan, 1987). The expert systems that Buchanan discusses have knowledge bases with hundreds of concepts in their vocabularies and contain thousands of rules. They solve problems with millions and tens of millions of possible solutions. It is not clear how to characterize the more complex problems that Chandrasekaran (Chandrasekaran and Mittal, 1983) mentions, nor the degree to which existing techniques will continue to scale as faster hardware becomes available.

3 Techniques for Building Expert Systems

There is a close connection between the techniques used to construct an expert system and the behavior exhibited by the program. The primary goal of this section is to analyze the dependencies between techniques and behaviors. Particular attention is given to previous arguments in the literature concerning limitations of existing techniques and proposals for new techniques that can provide the basis for more sophisticated behaviors.

The techniques that will be discussed fall into three classes. The first class concerns the *type* of knowledge to be used, such as causal, empirical, or first-principles knowledge. The second class pertains to *representing* and *reasoning* with this knowledge, such as the use of production rules and the explicit representation of control knowledge. Lastly, we consider the *amount* of knowledge a system contains. In what follows, each technique will be described in detail and we will consider how its use should contribute to the satisfaction of different behavioral requirements.

Our methodological comments at the beginning of Section 2 are relevant here. It is not clear why previous authors often call for the complete abandonment of existing techniques when little negative evidence exists to substantiate claims that a given behavior cannot be achieved using a certain technique. The desired behaviors may require only an improvement in degree of the capabilities of existing systems.

3.1 Causal Knowledge

There is a commonly held belief that first-generation expert systems did not use causal knowledge, and that such knowledge should supplant the use of empirical knowledge in expert systems. Exactly what constitutes causal knowledge is never made clear. Just because a knowledge base contains links labeled "causes" does not mean the program employs a well developed notion of causality. Philosophers have studied causality for hundreds of years without synthesizing a particularly coherent understanding of this complicated concept. In a much shorter time AI has, shall we say, not exceeded a proportionate contribution. We will consider philosophical discussions of causality and then examine the relation between causal knowledge and

expert systems.

Nagel lists four types of causal explanation in science: deductive explanation, probabilistic explanation, teleological explanation, and genetic explanation (Nagel, 1961) . He argues that some scientific laws have a causal basis, while others, such as the Boyle-Charles Gas Law, "simply asserts a certain concomitance in the variation of the specified attributes of a gas, and is therefore generally regarded as making no causal statement" (p22). He also notes that completely sufficient conditions for the occurrence of specific events are rarely if ever known.

In contrast, the philosopher Mackie centers his discussion of causality around what he terms an INUS condition, which is used to define a cause of an event:

> A is an INUS condition of a result P iff, for some X and some Y, $(A \wedge X) \vee Y$ is a necessary and sufficient condition of P, but A is not a sufficient condition of P and X is not a sufficient condition of P (Mackie, 1965).

Suppes analyzes five different conflicting intuitions about causality as developed by other philosophers, in terms of his own probabilistic theory of causality (Suppes, 1984). His theory includes the notion of a *genuine cause*, which is a prima facie cause that is not spurious. The terms *prima facie cause* and *spurious cause* are defined as follows.

> An event B is a prima facie cause of an event A if and only if (i) B occurs earlier than A, and (ii) the conditional probability of A occurring when B occurs is greater than the unconditional probability of A occurring.
>
> An event B is a spurious cause of A if and only if B is a prima facie cause of A, and there is a partition of events earlier than B such that the conditional probability of A, given B and any element of the partition, is the same as the conditional probability of A, given just the element of the partition.

A comparison the definitions made by these philosophers makes it clear that

many reasonable but incompatible notions of causality exist. It is thus rather confusing to read the suggestions below that future expert systems should include causal knowledge, and that existing expert systems do not include it, without any explication of what it means for a machine (or anything else) to have causal knowledge:

> There are domains where problem solving clearly relies on more than compiled experience. Other varieties of knowledge are involved, knowledge of structure and causal models (Davis, 1982) (p4).

> Surface systems ... have no underlying representation of such fundamental concepts as causality, intent, or basic physical principles... (Hart, 1982) (p12)

Examination of the philosophical literature reveals that causality is a much more complex and less well understood concept than most authors acknowledge. Thus, when AI authors provide few clues as to what they believe causal knowledge is, it is difficult to accept their claims that existing systems lack it, or that future systems will need it. In addition, these authors never make clear what specific behaviors this lack of causal knowledge prevents current programs from attaining, or will enable programs containing causal knowledge to attain.

What substance these claims do have stems from intuitive notions of causality, which we will use as the basis for further discussion. Our position is that it is not clear whether existing expert systems have causal knowledge or not. The discussion below suggests that if one accepts Suppes' definition of causality, MYCIN does have causal knowledge. An examination of the knowledge in the DENDRAL and CASNET programs suggests that their knowledge meets intuitive criteria of causality.

Let us consider the MYCIN steroids rule:

```
If    1) Infection requiring therapy is meningitis
      2) Only circumstantial evidence is available
      3) The type of the infection is bacterial
      4) The patient is receiving corticosteroids
```

```
Then  There is evidence that organisms causing
      the infection are klebsiella-pneumoniae (.2),
      e. coli (.4), or pseudomonas-aeruginosa (.1).
```

This rule is justified by the underlying knowledge that corticosteroids impair the body's ability to control organisms that normally reside within the body (Clancey, 1983). That is, corticosteroids cause the body to enter a state in which certain organisms are likely to proliferate to an abnormal degree. In this rule, corticosteroids fall under Suppes' definition of a prima facie cause of the infection because the conditional probability of infection by these organisms is higher when the patient is receiving corticosteroids than when they are not. That is, the rule links two events, A (infection by certain organisms), and B (administration of corticosteroids), where the conditional probability of A occurring when B occurs is greater than the unconditional probability of A occurring. Note that this definition of causality admits most empirical associations as causal knowledge. This position has been rejected by most previous authors in AI. Suppes' position may or may not be correct, but it is formulated with so much more detail and precision that it is worthy of serious consideration (which is beyond the scope of this paper).

It is also instructive to perform a thought experiment in which we transform the above rule and theory into these hypothetical rules:

```
If    1) The patient is receiving corticosteroids

Then  There is evidence that the patient's immune
      system is suppressed.
```

```
If    1) Infection requiring therapy is meningitis
      2) Only circumstantial evidence is available
      3) The type of the infection is bacterial
      4) The patient's immune system is suppressed

Then  There is evidence that organisms causing
```

```
the infection are klebsiella-pneumoniae (.2),
e. coli (.4),  or pseudomonas-aeruginosa (.1).
```

Here we have given MYCIN more knowledge than it had before, and apparently knowledge of a causal sort. We can imagine adding more and more rules which describe what factors suppress the body's immune system, and what cellular and chemical events are involved in that suppression. As MYCIN 's knowledge increases, it takes on a more and more causal character. But at no point was a new type of knowledge added to MYCIN ; we merely added more, increasingly detailed knowledge of a type which MYCIN already has.

One troubling aspect of the single MYCIN production rule above is that we know this rule is incomplete: MYCIN does not possess the medical knowledge which describes why a patient who receives corticosteroids is likely to be infected by the named organisms. Thus we feel uncomfortable saying MYCIN has causal knowledge because we know its knowledge is incomplete. This amounts to saying that a program only has causal knowledge if it knows all that experts know about a phenomenon. This is an unsatisfactory definition since it has the property that a program will cease to have causal knowledge of a phenomenon if experts acquire more knowledge about a phenomenon. This should not change the *type* of knowledge the program has, but only affects how correct and detailed we believe the knowledge to be.

Another interesting program to examine the CASNET program developed by Weiss *et al* (Weiss et al., 1978). CASNET diagnoses diseases related to glaucoma. To do so it employs three different *planes* of knowledge. The central plane is a causal model of disease processes in this domain, in the form of a graph. Nodes of the graph represent hypotheses about the disease process, and edges in the graph represent causal connections between these hypotheses, e.g., "*Cupping of the Optic Disc* causes *Glaucomatous Visual Field Loss*".

A second plane of knowledge contains clinical observations, which are connected to nodes in the causal plane. Thus clinical observations yield inferences about what disease processes are occurring in the patient. The third plane of knowledge specifies what disease processes in the causal plane are associated with what disease diagnoses. Thus, to diagnose a patient, CASNET uses clinical observations of the patient to determine what disease processes are occurring in the patient, and then finds what

diagnoses are associated with these disease processes.

TheDENDRAL program also uses causal knowledge. Its production rules describe the causal interactions between organic molecules and a mass spectrograph. These rules describe which chemical bond cleavages are likely to be caused by a molecule's passage through a mass spectrograph.

The most important conclusion of this section is that the term "causal knowledge" is very poorly defined within AI. In addition, a brief examination of several first generation expert systems suggests that they do contain some causal knowledge. Thus, the rather strong claim that existing expert systems contain no causal knowledge, but that it is essential to add this new sort of knowledge to future expert systems, should be deflated to the more reasonable claim that future expert systems will require more causal knowledge than existing systems have.

3.2 Empirical Knowledge

Genesereth has claimed that programs such asMYCIN andINTERNIST

> use 'shallow' theories of human pathophysiology in the form of 'rules' that associate symptoms with possible diseases. TheDART program contains no rules of this form. Instead, it works directly from a 'deep' theory consisting of information about intended structure ... and expected behavior" (Genesereth, 1984) (p412).

In a similar vein, Davis has claimed that when applied to building a digital circuit diagnostic system, the traditional *shallow* approach would lead to the creation of empirical associations such as the following, where B7 and AF2 are points within a circuit (Davis, 1982):

```
If      The signal is OK at B7  and
        The signal is blocked at AF2
Then    The signal is lost somewhere between B7 and AF2
```

Davis (Davis, 1982) quite correctly concludes that a better approach would be to write "a set of rules that tried to capture just the signal tracing skill, and had a separate description of structure." He summarizes:

> The point is simply that the Accepted Wisdom *focuses on the use of rules embodying empirical associations.* It does not offer us any tools for constructing structural descriptions of the sort we need, it does not offer us any techniques for using those descriptions to guide diagnosis, and perhaps even more important, *it does not even lead us to think in such terms* (Davis, 1982) (p18) [his italics].

And along these same lines, Chandrasekaran and Mittal say that

> Surface systems are at best a data base of pattern-decision pairs, with perhaps a simple control structure to navigate through the data base (Chandrasekaran and Mittal, 1983).

Davis and Genesereth are suggesting that we abandon the use of "rules embodying empirical associations" when constructing expert systems. This suggestion is unclear in several respects. Is it the method of knowledge representation and inference, the empirical associations, or both that are bad? What desired *behaviors* do they prevent us from attaining, and why?

It is unclear exactly what is an "empirical association". Consider an inference rule that states that a particular consequent should be inferred from a particular antecedent. Perhaps this rule is an empirical association if our only justification for writing the rule is that in the past we have empirically observed this antecedent and consequent to be associated with one another. A program which only uses empirical associations is one in which all inferences have this type of justification. Apparently, this is to be contrasted with rules with some other type of justification, e.g., if a scientific theory predicts that the antecedent will necessarily cause the consequent to occur, or when the antecedent implies the consequent by definition.

Using this definition, the quotes above are not accurate since many of MYCIN's rules are not simply empirical associations, but have some justification from medical

science, such as the steroids rule discussed in Section 3.1.

Perhaps then these authors have in mind a slightly different definition, namely that a program uses empirical associations if the program itself does not know the justifications of its inferences. MYCIN certainly does not know how medical science justifies the steroids rule. This knowledge might be useful to a program because it will allow the program to check the justification of a rule to be sure that the rule is applicable in the current situation, thus decreasing brittleness. An approach to providing explicit justifications for heuristic rules is described in (Smith et al., 1985).

A quest to produce an expert system free of empirical associations can never succeed since the program's justification structures must *bottom out somewhere.* If every justification is supported by another justification then the program must employ either an infinite set of justifications or a circular set. The unjustified rules in a program with neither of these properties must be empirical associations. Thus every finite program that is free of circular reasoning must rest upon empirical associations. The only alternative is resting on unjustified assumptions or postulates.

That empirical associations are essential should not be surprising since all of our scientific *causal* knowledge ultimately rests on experimental data that consists of *empirical associations.* The inductive inference from which scientific theories are constructed is even less trustworthy. If empirical associations are the basis of all of science, perhaps they are not so bad after all.

One way to act upon the advice above to avoid using empirical associations is to provide a program with as many justifications as possible for the rules that it uses. But we must recognize that these justifications will ultimately contain many empirical associations.

3.3 First-Principles Knowledge

It has been suggested that existing expert systems do not use first principles, and that their use will be important to future expert systems:

> Reasoning from First Principles offers the possibility of dealing with novel faults (Davis, 1984).

> At the extremes, a surface system directly associates input states with actions, whereas a deep system makes deductions from a compact collection of fundamental principles (Hart, 1982) (p12).

> The major intuition behind the feeling that expert systems should have deep models is the observation that often even human experts resort to *first principles* when confronted with an especially knotty problem. Also, there is the empirical observation that a human expert who cannot explain the basis for his reasoning by appropriate reference to the deeper principles of his field will have credibility problems... (Chandrasekaran and Mittal, 1983).

These authors seem to have the intuition that the behavioral goals of solving novel problems – such as removing brittleness and solving more complex problems – can be satisfied by giving programs knowledge of and the ability to reason from first principles. The key characteristics of these principles appear to be general applicability but weak ability to aid in solving specific problems. Generality is what makes them superior to the *non-first principles* contained by the program: first principles are sufficiently general to apply to some of the novel, difficult, and peripheral problems for which the non-first principles fail. It is on these problems that the system "falls back on first principles".

But non-first principles must have advantages over first principles since first principles are apparently applied after the former have failed, not in place of the former. That is, *first* principles are tried *last*, *non-first* principles are tried *first*. The non-first principles could have several advantages. First principles might be slower than non-first principles: a very general problem solving method is expected to be slower than methods that have been tailored to particular problems. First principles could be more difficult to apply to a specific problem – they may be difficult to operationalize. Sometimes it may not be possible to operationalize first principles at all – they might not apply to all problems in a domain. Finally, first principles might

produce incorrect answers, either because they are overly general or because they are not accurate.

First principles might differ from non-first principles in terms of speed, completeness, and correctness. Hart (Hart, 1982) has the intuition that first principles should be *compact* and *fundamental.* It is easy to think of domains that do not have compact first principles. Medicine has a huge, incomplete, inaccurate set of first principles (much of biology), education has even more incomplete and inaccurate first principles, quantum mechanics has very complex first principles; and digital circuit diagnosis appears to have a complete, correct, and small set of first principles.

One might define first principles as the lowest level of knowledge usually employed by a professional community, i.e., the lowest level of knowledge to which an expert must usually revert when solving problems in his or her domain. Thus, first principles for molecular biologists are the laws of chemistry, and first principles for teachers include theories of how children learn.

The large variation in the form of first principles makes them extremely difficult to distinguish from non-first principles. We have characterized first principles as being general and some combination of slow, incomplete and incorrect. All these adjectives can easily be applied to the principles built into existing expert systems – the latter have some degree of generality and certainly have limitations of speed, completeness and correctness. Given the range in the expected properties for first principles in the domains above it is not at all clear when we can characterize a set of principles as *first principles* and when we cannot.

This has two implications. First, if first principles are so hard to distinguish from the principles in existing expert systems, it is not obvious why the techniques used to encode these existing principles (e.g., production rules) would not suffice to encode first principles.

Second, it would appear that we can operationalize the advice to "use first principles to build deep systems" into the advice to construct an expert system that can attempt to apply more than one problem solving method to a given problem. The fastest, most specific, most complete, most correct method is attempted first. Thus, the meaning of "fall back on first principles" is simply to bring an *additional*

knowledge to bear on a problem, not a fundamentally *new kind* of method. Attacking a problem with multiple knowledge sources will improve a problem solver's behavior by contributing to the solution of novel, difficult, and peripheral problems. appears to be excellent advice, that in essence instructs us to build This appears to be excellent advice, that in essence instructs us to build programs with more knowledge. Notice that this differs from the original interpretation of this advice, which viewed first principles as a new, special kind of knowledge which existing programs do not have.

3.4 Structural and Functional Knowledge

It has been argued that future expert systems will require knowledge of the structure and function of physical systems, and that this is a type of knowledge that the first-generation expert systems did not have or use. The quotations by Davis and Genesereth in Section 3.1 are examples of such statements. This case is very similar to that of causal knowledge: we assert that existing expert systems do in fact contain some structural and functional knowledge, but their performance will probably be improved if they are given more of this type of knowledge.

The DENDRAL program (Lindsay et al., 1980) has explicit knowledge concerning the allowable chemical structures that complex organic molecules can assume, and can generate all the potential chemical structures that can potentially be formed when a complex molecule is heated within a mass spectrograph. The operation of the DENDRAL program can be viewed as follows. DENDRAL starts with a deep first-principles or structure-function model of chemistry, called CONGEN. Given this deep theoretical model, and given empirical knowledge of the behavior or molecules in a mass spectrogram as a set of classified training instances, DENDRAL synthesizes a set of shallow production rules that can interpret mass spectrograms.

The R1 program configures Digital VAX computers from customer orders (McDermott, 1982). It has extensive knowledge of the structural components of VAXes, and of the properties of these components such as their power consumption and the ways in which they may be interconnected.

In certain domains, a structure-function model is not available. For example, in the medical domain of the MYCIN program, the pathways of most of the processes

are unknown to medical science. Even simple knowledge, such as the mechanism by which an infection causes a fever, is unknown. Hence the reasoning of MYCIN and physicians in the domain of meningitis proceeds from heuristic associational knowledge of the behaviors (symptoms) that result from different structures (the human body combined with different bacteria).

It seems very plausible that there exist problems whose solutions require significantly more knowledge of structure and function than existing expert systems have. But it is not true that these systems do not use knowledge of structure or function.

3.5 Amount of Knowledge

As stated earlier, our central thesis is that the desired behavioral requirements for future expert systems will be realized by providing them with larger amounts of knowledge rather than new types of knowledge which they do not currently employ. Techniques for managing and utilizing this knowledge will also be important as we discuss in later sections of this paper.

The following test has been proposed by which one may decide whether one program is deeper than another.

> Consider two models of expertise M and M'. We will say that M' is *deeper-than* M if there exists some implicit knowledge in M which is explicitly represented or computed in M'" (Klein and Finin, 1987).

The authors of this test acknowledge that notions such as implicit and explicit knowledge are left to intuition, so it can be difficult to apply this predicate in many cases.

The reason to build programs that contain more knowledge is so they will solve problems whose solution requires more knowledge. Consideration of the amount of knowledge necessary to solve problems in different domains provides an explanation of both the capabilities and limitations of existing expert systems. The behavior of many expert systems is described by the graph in Figure 1. It suggests how the

competence of a program changes over time as it contains more and more knowledge. This accurately characterizes the experience of adding more and more rules for R1 program via manual knowledge acquisition when failures were encountered (McDermott, 1982). A similar analysis is given by Feigenbaum and Lenat (Feigenbaum and Lenat, 1987).

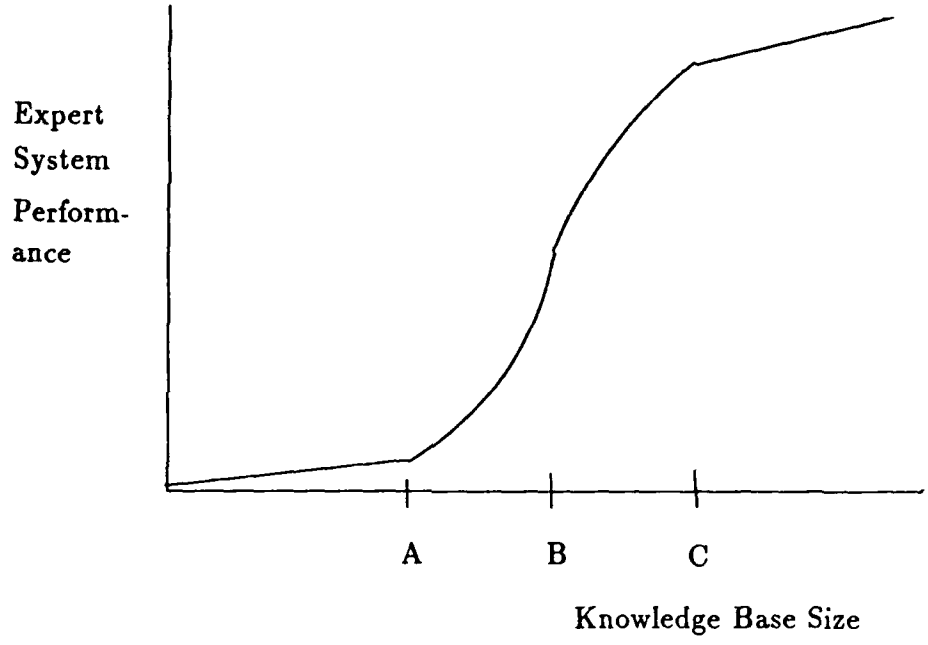


Figure 1: This figure shows the relationship between the size of an expert system's knowledge base and the sophistication of the problem-solving performance.

Two properties of this curve are important: the fact that it increases monotonically, and the fact that its derivative (the marginal utility of knowledge) starts out small, grows significantly larger, and then decreases again. The behavior of the curves derivative appears to be do to the existence of a relatively small "core set" of knowledge which is required to solve a relatively large set of problems in the domain, with larger amounts of more esoteric knowledge required to round out the system. Between points A and B, new knowledge interacts synergistically with old to increase competence tremendously. But after point B, the marginal utility of knowledge begins to decrease, and most core knowledge for the domain has been captured by the time point C is reached.

Different problem domains would exhibit curves with the same shape, but with

different scaling along the axes, e.g., a very complex domain would scale the curve outward to the right, requiring more knowledge to exhibit the same degree of competence. Imagine building expert systems of roughly the same size in many different domains. Their performance would vary according to the scale of the curve in that domain. For example, MYCIN probably lies near point C: its validation studies have shown that it is able to solve many problems within its domain. After a system has reached this level of competence, for each new case that is unable to be solved, more and more rules will be added to the knowledge base that are each relevant to fewer and fewer cases. But if a system of roughly MYCIN's size is to be built in a much more complex domain, the system would be closer to point A, where it exhibits poor performance, but a high marginal utility of knowledge. This situation has often led various authors to hypothesize the explanations for the shortcomings of existing expert systems such as those discussed earlier in this paper. It may be that the limitations of these systems are due not to the types of knowledge the systems employ, but to the complexity of the different problem domains: different domains require different amounts of knowledge for a given level of problem-solving competence.

Figure 1 is of course a simplification. The curve might start out completely flat because some critical mass is needed before any problems can be solved. And the curve might actually decrease at the right because the program is unable to properly employ a huge knowledge base. Curves of slightly different shapes may result from adding different pieces of knowledge in variable order.

3.6 Production Rules

Production rules have been described as an inappropriate basis for constructing programs that exhibit a number of the behaviors that have been previously described (Davis, 1984; Genesereth, 1984).

First, let us clarify this hypothesis. Certainly the claim is not being made that production rules cannot in principle be used to achieve these behaviors. Since production rules have been proved Turing-equivalent (Brainerd and Landweber, 1974) this claim would imply that the behavioral goal could never be met by a computer program, so instead, the claim must be that production rules are not a good engineering tool for this task. Claims of this form are certainly reasonable; such concerns

motivate the design of new programming languages, for example. However, most authors never make clear what the engineering limitations of production rules are. Thus it is difficult to comprehend their arguments and to see where future research should be directed.

The quotations in Section 3.1 by Davis and Genesereth discuss the degree to which production rules are suited to representing the structure and behavior of digital circuits. Genesereth (Genesereth, 1984) contrasts production rule representations used by expert systems with representations used by the DART program such as the description of an AND gate shown below:

```
(If  (AND (ANDG d)
          (VAL (IN 1 d) t ON)
     (VAL (IN 2 d) t ON))
Then (VAL (OUT 1 d) t ON)))

(If  (AND (ANDG d)
     (VAL (IN 1 d) t OFF))
Then (VAL (OUT 1 d) t OFF)))

(If  (AND (ANDG d)
     (VAL (IN 2 d) t OFF))
Then (VAL (OUT 1 d) t OFF)))
```

But Genesereth's description *is in fact a set of production rules* which represent the behavior of an AND gate. These rules directly associate the possible inputs of an AND gate with the outputs it would generate. Thus production rules are able to encode structural and functional knowledge about a device, contrary to Davis and Genesereth's assertions.

De Kleer and Brown (DeKleer and Brown, 1984) hold the view that constraints should be used to express device behavior instead of production rules. They offer three arguments in support of this view.

The first argument is that constraints can express behavior more succinctly than production rules. Imagine that we wish to describe the behavior of a pipe by

stating that its input pressure must match its output pressure. DeKleer (DeKleer and Brown, 1984) states that within the {-1, 0, +1} value space used to qualitatively represent numerical processes, this could be done with one constraint:

$$P_{in} = P_{out}$$

but this same expression would require the use of six production rules:

(IF (EQUAL P_{in} -1) THEN (SET P_{out} -1))
(IF (EQUAL P_{in} 0) THEN (SET P_{out} 0))
(IF (EQUAL P_{in} 1) THEN (SET P_{out} 1))
(IF (EQUAL P_{out} -1) THEN (SET P_{in} -1))
(IF (EQUAL P_{out} 0) THEN (SET P_{in} 0))
(IF (EQUAL P_{out} 1) THEN (SET P_{in} 1))

Now it is fairly trivial to condense this set of rules by the use of variables to:

(IF (EQUAL $P_{in}$$X) THEN (SET $P_{out}$$X))

(IF (EQUAL $P_{out}$$X) THEN (SET P_{in} $X))

The constraint is still slightly more concise here, because constraints are *bidirectional*, whereas rules are *unidirectional*. While rules can be invoked within both a forward chaining and a backward chaining interpreter, a valid inference is assumed to occur only in the forward direction; backward chaining can create only a goal, not a conclusion. Put another way, the rule $A \supset B$ allows a system to conclude that B is true when it knows A is true, but not that A is false when B is known to be false. However, constraints will cause problems when we wish to express unidirectional relations, or when we have different degrees of belief in inferences in the two directions. So the following constraint is *too* expressive (we believe the forward but not necessarily the backward inference):

light-switch-off = light-bulb-off

The second argument offered in favor of constraints in (DeKleer and Brown, 1984) is that the *if-then* form of production rules "falsely implies the passage of time" between the test and action. Perhaps this point depends on exactly who is reading a given rule; it is not clear which interpretation is the correct one. But symmetrically, perhaps constraints falsely imply instantaneous linkages within a device where in fact it takes time for changes to propagate. In an event, it is quite possible to modify production rules to include a time parameter which allows one to explicitly indicate whether or not time is passing:

IF (EQUAL P_{in}\$X T_1) THEN (SET P_{out}\$X $T_{1+\epsilon}$)

The third argument is more an indictment of the technique of forward simulation than of production rules per se. DeKleer and Brown describe an example in which two pipes are connected, with the pressure rising at one end and held constant at the other end. They find that if one uses production rules to represent local information relating pressure and flow, and then reasons in a traditional forward direction to try to derive the pressure at the joint between the pipes, this solution method fails. The reason given is that this problem cannot be solved by simply propagating values through the constraints which describe these pipes (i.e., by forward chaining through production rules). The pressure at the joint must be derived by assuming every possible value for this pressure, and then determining if any constraints are violated. The assumed pressure which does not yield a contradiction is the solution. They conclude that: "constraints can support both imperative interpretations (they can be executed) and assertional interpretations (i.e., they can be reasoned over)".

This limitation is hardly related to the use of production rules. Production rules can be both executed and reasoned over just as constraints can be. If DeKleer and Brown had attempted to utilize constraints in a forward qualitative simulation they would have just the same problem as with production rules. It is common for AI programs to manipulate a set of production rules in several ways: MYCIN's rules were used for diagnosis, for explanation, and for interactive transfer of expertise in the TEIRESIAS program (Davis, 1982).

Another possibility is that these authors are criticizing the use of production *systems* rather than production *rules*, i.e., the use of a fixed forward or backward

chaining interpreter to evaluate these rules. Davis and Genesereth use techniques called Constraint Suspension and Resolution Residue, respectively, in their circuit diagnosis programs. It is reasonable to assert that different tasks (e.g., diagnosis versus prediction) will require different reasoning mechanisms. But as discussed in the previous paragraph, single sets of production rules have been used by multiple reasoners since the days of MYCIN.

Thus, we see that production rules do not have a number of the limitations which authors have claimed. Previous statements by Davis, Hart and Genesereth have also suggested that production rules are unable to represent causal knowledge, knowledge of structure and function, or first principles. We have shown in earlier sections that systems such as MYCIN and DENDRAL, which are built from production rules, do contain this type of knowledge.

3.7 Multi-Level Domain Models

Davis examines the reasoning a human expert might use to track down a fault in a computer system (Davis, 1984) (page 14-19). In this example the expert is given a gross description of the machine's functionality: the system boots and responds properly at the console, but user terminals are dead. On this basis the expert rules out a number of the machine's components at the Processor/Memory/Switch (PMS) level: the CPU, the system disk, and the bus between them. Davis then focuses on the terminal bus and the terminals themselves and rules out the latter. He probes the internal state of the bus the terminals are connected to, focusing on increasingly detailed descriptions of the hardware. Eventually he is able to determine that the bus interface is dropping a bit.

This example involves reasoning about the internal structure of complex devices at several levels of detail. The descriptions at each level represent the structure and function of the device with different degrees of abstraction. Each description is fairly independent of the others in that it can accept diagnostic information that is compatible with its level of abstraction, and compute diagnostic hypotheses and tests at that level of abstraction. But some connections between the descriptions must exist as well, to allow a global diagnostic procedure to home in on a precise diagnosis using more and more detailed descriptions in more and more restricted regions of the

device. We call such descriptions *multi-level domain models*[1].

This organization of knowledge is a very fruitful avenue for future research because it appears to be relevant to producing a number of the desired behaviors. One advantage of this approach is faster problem solving. A system may be able to achieve satisfactory performance if it reasons at an abstract level for problems whose solution does not require all the system's knowledge. Or it could time-share problem-solving at several levels of detail simultaneously to produce the most detailed solution possible within unpredictable time constraints.

Another possibility provided by the framework is validation of the scope of problem solving. Perhaps abstract levels of knowledge could be used for problem solving, while underlying levels could be used to determine if the abstract levels are applicable to a given problem. An underlying theory consisting of Newton's laws might contain qualifications such as: only applicable to objects moving slowly compared to the speed of light, and in weak gravitational fields. These laws might be compiled to produce rules for solving problems involving bouncing balls. But the qualifications at the underlying levels could be checked before application of the compiled level. Yet another advantage of multiple levels relates to the production of explanations. Abstract levels of a multi-level domain model could be used to generate concise, general explanations, while lower levels could generate detailed, focused explanations.

Several first generation expert systems used descriptions with only some of the properties above. Their domain models were structured into descriptions at several levels of detail. But, they were not distinct in the sense described above because generally all levels of detail had to be employed when solving a given problem. For example, the HEARSAY speech understanding system (Erman et al., 1980) did model speech understanding at several different levels of detail. But the input signal always arrived at the lowest level of detail and problem solving propagated the answer up through the system to produce outputs at the sentence level (in fact processing was not strictly bottom-up). Similarly, Patil's ABEL system contained multiple descriptions at different levels of abstraction in the medical domain of electrolyte disorders (Patil et al., 1981), and the HELIOS digital circuit simulator models digital circuits at several

[1] These are to be distinguished from systems that contain a level of domain knowledge plus one or more levels of control knowledge.

interlocking levels of detail (Brown et al., 1983; Foyster, 1984).

A promising approach relates to deriving a shallow reasoning system from a deep causal model, such as described in (Pearce, 1988). This shallow reasoning system is superior in terms of time and space complexity measures, and can also be guaranteed to be complete. The completeness of the model is of course with respect to the causal model, which may or may not accurately mirror the real world.

In summary, while first generation expert systems include descriptions with some of the attributes of multi-level domain models, further research is required to endow future systems with the behaviors we have here described.

3.8 Reasoning about Real-Valued State Variables

There is a large class of problems whose solutions require detailed reasoning about the relationships between real-valued state variables of a system. For example, when one constructs a generative model of a device, one may need to combine the pressures, concentrations, temperatures, masses, accelerations, and/or voltages of its components to compute the aggregate behavior of the device. Human experts are often able to reason about such dimensions of a system when only incomplete information is available about the values of these variables and the relationships among them.

Authors of existing expert systems have largely side-stepped these issues by tackling classes of problems where this type of reasoning is either not required, or where very simple types of reasoning will suffice, e.g., in MYCIN's domain.

In the past few years, researchers in the AI subfield of qualitative reasoning have developed new techniques for reasoning about complex interactions between real-valued state variables in the presence of incomplete information and in the presence of complicating factors such as feedback. New qualitative representations have been developed for both the values of state variables and for the interactions between them and for predicting the potential behaviors of such a system (Forbus, 1984; Karp and Friedland, 1988; DeKleer and Brown, 1984; Kuipers, 1985; Kuipers, 1986; Simmons, 1986). Significant progress has been made in this field, but it is not possible to construct programs whose understanding of a complex device approaches that of a

human expert.

3.9 Explicit Representation of Control Knowledge

Another technique which has been explored in the development of expert systems is the separation of the program's control knowledge from its domain knowledge, and the explicit representation of this control knowledge. This was actually a major goal in the construction of MYCIN, wherein the expert system was viewed as consisting of a domain knowledge base and an inference engine that accomplished control by backward chaining of production rules.

Important efforts that have investigated the separation of control knowledge from domain knowledge include the NEOMYCIN program, the BB1 blackboard architecture, and the MRS logic programming language. NEOMYCIN separates knowledge of medical domain knowledge of meningitis infections from strategy knowledge for performing medical diagnosis (Clancey, 1984). BB1 allows a programmer to separate problem solving operators in a domain from control strategies and heuristics that guide the application of these operators (Hayes-Roth, 1985). MRS provides metalevel facilities to control inference in a Prolog-like logic programming language (Russell, 1985).

There are many advantages to a clean separation of domain and control knowledge. First, it facilitates using the same knowledge for different purposes, such as diagnosis, design, teaching, and explanation. Separate inference procedures can be constructed for each of these tasks, and each of them can use the same domain knowledge base. Second, an explicit representation of the control knowledge facilitates the construction of programs that can inspect and reason about the control knowledge. MRS allows dynamic determination of the order in which clauses of a conjunctive rule are executed, using its metalevel control facilities. Third, it simplifies the problem of knowledge acquisition. When a knowledge acquisition program modifies the knowledge structures of a program, there is less chance that the modifications will have unforeseen side effects. Fourth, it can lead to cleaner, more structured explanations that differentiate between the domain operators that are currently being applied, and the control strategy that has caused them to be applied at a particular time. Lastly, explicit representation of control knowledge facilitates optimizing control for a specific

task. This produces faster performance in some cases.

A complete separation of control and domain knowledge, and the explicit representation of control knowledge, can be viewed as long-term research goals. The NEOMYCIN, BB1, and MRS systems represent significant steps in the achievement of this goal. However, part of the lesson from constructing these systems has been the discovery of ways in which there is an incomplete separation of domain and control knowledge, and a non-explicit representation of control knowledge.

3.10 Compilation

Within computer science, the term "compilation" usually refers to transforming the representation of a program to increase its efficiency or to make the program operational on a particular hardware architecture. Within artificial intelligence, compilation also refers to the process of transforming a program to increase its efficiency, although often the program is to be optimized for a particular class of problems. The LEX program, for example, transforms its control knowledge (which is represented as preconditions for integration operators) in the process of solving symbolic integration problems. Another good example is the SOAR program in which the principal learning method is *chunking* preconditions for problem-solving operators in the course of solving problems (Laird et al., 1987).

The principal method of obtaining a compiled domain model for an expert system has been to extract it from an human expert. This is the method that was used to obtain the compiled domain model that is used by the MYCIN program. Another approach to obtaining a compiled domain model is to extract a deep domain model from a human expert, and then compile this deep domain model into a shallow domain model. This has two advantages. The first advantage is that a deep domain model can often be created from books or from documentation of a domain by someone who is not a domain expert. Another advantage is that a single deep domain model can sometimes be compiled into many different shallow domain models, where each different shallow domain model is created for a different class of problems, such as design and diagnosis.

Compilation almost always yields an improvement in the space-time complexity

of a program over the uncompiled version of the program. Another potential advantage with respect to expert systems is shorter explanations, since the information is compressed. Note that information is lost during the process of compilation of a deep domain model into a shallow one, and so the process is not reversible.

There have been quite a few research efforts in the direction of creating a shallow domain model from a deep domain model. The usual approach is to synthesize a functional model of a device into an associational model, for purposes of diagnostic problem solving (Chandrasekaran and Mittal, 1983).

One of the advantages that is often claimed for deep domain models is that they provide a fallback when a shallow domain model proves inadequate. However, a shallow model may be created from a deep model in such a way that this will not be true, as described in the following conjecture (Chandrasekaran and Mittal, 1983). They hypothesize that it may be possible to compile a shallow domain model for a particular goal from a deep domain model, such that all the problems that the deep model is able to solve can be solved by the shallow model.

> Between the extremes of a data base of patterns on one hand and representations of deep knowledge (in whatever form) on the other, there exists a knowledge and problem-solving structure which (1) has all the relevant deep knowledge "compiled" into it in such a way that it can handle all the diagnostic problems that the deep knowledge is supposed to handle if it is explicitly represented and used in problem-solving; and (2) will solve the diagnostic problems more efficiently; but (3) it cannot solve other types of problems - i.e. problems which are not diagnostic in nature - that the deep knowledge structure potentially could handle.

Even if a shallow domain model that is derived from a deep domain model can solve all the problems that can be solved by the deep model, the complexity of the resulting shallow model may make this approach not worthwhile. In such a case, the correct approach would be to compile a shallow model that would give fast solutions to most problems, and use the deep model for those cases in which the shallow domain model is inadequate.

The concept of *encapsulation* provides a nice distinction to elucidate the differ-

ence between compiled and deep knowledge (Simmons, 1988). Compiled knowledge encapsulates the interactions that occur between knowledge elements; elements that interact are represented as different clauses in the same compiled rule.

There are many open questions that relate to compilation. For example, given a change to a deep domain model, can the shallow domain model be incrementally changed? There is a lot of work that relates to quantifying the space-time complexity advantages of compilation. For example, might it be worthwhile to have several different compiled versions, and always use the most efficient version that is sufficiently detailed for the current problem?

4 Summary

The first generation of expert systems (1972-1981) is often described as using only shallow methods of representation and inference. These expert systems are then dismissed on the grounds that use of these methods prevents them from achieving problem-solving behaviors that expert systems should possess. This paper analyzed the dependencies between behaviors and techniques to determine what existing techniques are limiting the behaviors of expert systems, and what new techniques are likely to extend their capabilities. This analysis conflicted with those of other authors in several ways. Some techniques and behaviors that others have discussed were poorly defined; in other cases the links between techniques and behaviors were either not specified or not well justified. And some proposed new techniques for future systems have in fact been used by previous systems.

The past inaccurate characterizations of first-generation expert are surprisingly widespread, which has had two deleterious effects. First, so-called deep methods are often improperly viewed as an an alternative to shallow methods, rather than as an augmentation. Second, there is an overemphasis on developing new methods to address the behavioral shortcomings of expert systems, as opposed to investigating existing techniques in more detail to extend their power.

This paper first discussed a set of behaviors which are important for expert systems to exhibit. The behaviors discussed were the following:

Expert systems must be able to *explain* their problem solving behavior. These explanations should be tailored to what the user knows so that they are neither too detailed nor too abstract.

The problem-solving performance of an expert system should *degrade gracefully* as the problems presented to the system depart from the system's domain of expertise. While early expert systems such as MYCIN were in fact able to detect that some problems were outside their expertise, very little research has been done since to determine how difficult such limit detection is in general.

As the functionality of expert systems has increased, their *speed* has often decreased. Future systems must not only solve complex problems, they must solve them

within reasonable resource limitations.

Some authors have suggested that the first generation of expert systems are unable to *solve novel problems*, but that future systems should have this capability. Our analysis shows that it is hard to define what novelty is, but one can consider novel problems as those whose solution the programmer has not anticipated. We concluded that whether the programmer constructs a system using generative or enumerative candidate descriptions, he or she must always anticipate what *classes* of problems the system will encounter.

To solve some classes of problems, the problem solver must reason about *real-valued state variables* such as pressures and voltages. Standard numerical techniques are not appropriate when only imprecise, qualitative values are available for these variables, which is sometimes the case for problems that expert systems should solve.

Just as experts can *employ their knowledge for multiple purposes*, it is desirable for expert systems to use their knowledge in many ways such as diagnosis, teaching, and the acquisition of additional knowledge. Knowledge representation methods are an important key for this ability, and those used by NEOMYCIN and associated programs allowed a single knowledge base to be used for the multiple purposes listed above.

Finally, some authors have suggested that future expert systems will *solve more complex problems* than first generation systems. These suggestions do not identify what techniques are limiting the complexity of problems that existing systems can solve, and in general it is difficult to be precise about this issue because the field has few methods for characterizing the complexity of a problem domain.

The next section of the paper considered various techniques for constructing expert systems that might endow them with the behaviors above. First, the use of certain *types* of knowledge in future systems was considered. Several authors have suggested that the first generation of expert systems lack these types of knowledge, but that future systems will require them.

Consideration of *causal knowledge* revealed that it is not at all clear just how one distinguishes it from other types of knowledge; philosophers present many possible definitions of causality. But, if we proceed from an intuitive notion of causality, it is clear that early expert systems such as DENDRAL did possess causal knowledge.

Similar remarks apply to *structure-function knowledge* of a device (although these are easier to define): many early systems did have this sort of knowledge.

Knowledge of *first principles* was considered next. We showed that the characteristics of first principles knowledge varied tremendously between application domains. Thus, we could only operationalize the method of "falling back on first principles" as advice to bring a second problem solving method to bear on a problem. So first principles are not actually a *type* of knowledge, but merely an additional source of knowledge whose properties vary in different domains.

The next section considered the use of *empirical associations*, which other authors have suggested should not be used in future systems. "Empirical associations" proved to be yet another fuzzy term, and analysis showed that it is hard to imagine building systems where there is no important role for empirical knowledge, especially since all of our theoretical knowledge ultimately rests on empirical experience.

Next we considered various ways of *structuring* knowledge in expert systems.

Several criticisms of *production rules* were considered and found to have little substance. Just as constraints (a proposed alternative) can be reasoned over for several purposes, so too can production rules. Production rules are relevant to almost all behaviors of expert systems since they provide fundamental reasoning and representational building blocks.

Multi-level domain models are relatively independent models of a domain at different levels of abstraction. They seem relevant to providing better explanations, increased problem solving speed, and more sophisticated problem solving. Further research will be required to provide expert systems with such models.

Qualitative reasoning about real-valued state variables is a relatively new area of research; previous expert systems had no techniques to solve problems with qualitative knowledge of real-valued state variables.

Separation of domain from control knowledge has improved explanations, problem solving speed, and multi-purpose problem solving in existing expert systems. It is likely to be important in future systems as well.

Finally, including larger amounts of knowledge in future expert systems will likely be a key method for improving both their problem solving sophistication and the grace with which their performance degrades on peripheral problems. But determining how to structure large quantities of knowledge for fast use, understandable explanations, and the solution of multiple types of problems will present significant challenges which future researchers must address.

Acknowledgements

Many people have provided us with valuable comments on drafts of this paper. We thank them collectively, and in particular thank Bruce Buchanan, Jim Bennett, Robert Engelmore, Haym Hirsh, Peter Friedland, Phyllis Koton, Paul Rosenbloom, Stuart Russell, Narinder Singh, Michael Walker, and Marianne Winslett.

This work was supported by ONR grant N00014-88K0124, NSF grant MCS83-10236, NIH grant RR-00785, and DARPA contract N00039-83-C-0136.

References

Brainerd, W. S. and Landweber, H. L. (1974). *Theory of Computation.* John Wiley.

Brown, H., Tong, C., and Foyster, G. (1983). Palladio: An exploratory environment for circuit design. *IEEE Computer*, 16(12).

Buchanan, B. (1987). Artificial intelligence as an experimental science. Technical Report KSL-87-03, Stanford University.

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.

Chandrasekaran, B. and Mittal, S. (1983). Deep versus compiled knowledge approaches to diagnostic problem-solving. *Int. J. Man-Machine Studies*, pages 425–436.

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. (1986). From guidon to neomycin to heracles in twenty short lessons. *AI Magazine*, 7:40–60.

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*, pages 227–490. New York: McGraw-Hill.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1-3):347–410.

DeKleer, J. and Brown, J. S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, 24(1-3):7–84.

DeKleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130.

Erman, L., Hayes-Roth, F., Lesser, V., and Reddy, D. (1980). The hearsay-ii speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253.

Feigenbaum, E. and Lenat, D. (1987). On the thresholds of knowledge. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1173–1182.

Feigenbaum, E. A. (1977). The art of artificial intelligence: Themes and case studies in knowledge engineering. In *Proceedings of the 1977 IJCAI*, pages 1014–1029, Cambridge, Mass.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Forbus, K. D. (1988). Intelligent computer-aided engineering. *AI Magazine*, 9:23–36.

Foyster, G. (1984). HELIOS: user's manual. Knowledge Systems Laboratory Report KSL-84-34, Stanford University, Stanford, CA.

Genesereth, M. R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411–436.

Hart, P. (1982). Directions for AI for the eighties. *Sigart Newsletter*, 79.

Page 42 not included in doc.

Pople, H. E. (1982). Heuristic methods for imposing structure on ill-structured problems: The structuring of medical diagnostics. In Szolovits, P., editor, *Artificial Intelligence in Medicine*, pages 119–190. Boulder: Westview Press.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96.

Russell, S. (1985). The compleat guide to MRS. Knowledge Systems Laboratory Report KSL-85-108, Stanford University, Stanford, CA.

Simmons, R. (1986). Commonsense arithmetic reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 118–124.

Simmons, R. (1988). Using associational and causal reasoning to achieve efficiency and robustness in problem solving. In *Proceedings of the International Association for Mathematics and Computers in Simulation (IMACS).*

Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. G. (1985). Representation and use of explicit justifications for knowledge base refinement. In *Proceedings of the 1985 IJCAI*, pages 673–680, Los Angeles, CA.

Suppes, P. (1984). *Probabilistic Metaphysics.* Blackwell.

Weiss, S., Kulikowski, C., Amarel, S., and Safir, A. (1978). A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11(1–2):145–172.

Wilkins, D. C. (1988a). Apprenticeship learning techniques for knowledge based systems. Knowledge Systems Laboratory Report KSL-88-14, Stanford University. 155 pages.

Wilkins, D. C. (1988b). Knowledge base refinement using apprenticeship learning techniques. In *Proceeedings of the Seventh National Conference on Artificial Intelligence*, pages 646–651, Minneapolis, MN.

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Ms. Lisa B. Achille
Code 5530
Naval Research Lab
Overlook Drive
Washington, DC 20375-5000

Dr. Edith Ackermann
Media Laboratory
E15-311 20 Ames Street
Cambridge, MA 02139

Dr. Philip Ackerman
Dept. of Psychology
University of Minnesota
75 East River road
N128 Elliott Hall
Minneapolis, MN 55455

Dr. Beth Adelson
Department of Computer Science
Tufts University
Medford, MA 02155

Technical Document Center
AFHRL/LRS-TDC
Wright-Patterson AFB
OH 45433-6503

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Robert M. Aiken
Computer Science Department
038-24
Temple University
Philadelphia, PA 19122

Dr. Jan Aikins
AION Corporation
101 University
Palo Alto, CA 94301

Dr. Saul Amarel
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Mr. Tejwansh S. Anand
Philips Laboratories
345 Scarborough Road
Briarcliff Manor
New York, NY 10520

Dr. James Anderson
Brown University
Department of Psychology
Providence, RI 02912

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Thomas H. Anderson
Center for the Study of Reading
174 Children's Research Center
51 Gerty Drive
Champaign, IL 61820

Dr. Stephen J. Andriole, Chairman
Department of Information Systems
and Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. John Annett
University of Warwick
Department of Psychology
Coventry CV4 7AL
ENGLAND

Dr. Edward Atkins
Code 61Z1210
Naval Sea Systems Command
Washington, DC 20362-5101

Dr. Michael E. Atwood
NYNEX
AI Laboratory
500 Westchester Avenue
White Plains, NY 10604

Dr. Patricia Baggett
School of Education
610 E. University
University of Michigan
Ann Arbor, MI 48109-1259

Dr. Bruce W. Ballard
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Dr. Donald E. Bamber
Code 448
Naval Ocean Systems Center
San Diego, CA 92152-5000

Dr. Harold Bamford
National Science Foundation
1800 G Street, N.W.
Washington, DC 20550

Dr. Ranan Banerji
Dept. of Mathematics and CS
St. Joseph's University
5600 City Avenue
Philadelphia, PA 19131

Dipartimento di Psicologia
Via della Pergola 48
50121 Firenze
ITALY

Dr. Marie A. Bienkowski
223 Ravenswood Ave. EK337
SRI International
Menlo Park, CA 94025

Dr. Gautam Biswas
Department of Computer Science
Box 1688, Station B
Vanderbilt University
Nashville, TN 37235

Dr. John Black
Teachers College, Box 8
Columbia University
525 West 120th Street
New York, NY 10027

Dr. Daniel G. Bobrow
Intelligent Systems Laboratory
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Dr. Deborah A. Boehm-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Sue Bogner
Army Research Institute
ATTN: PERI-SF
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. C. Alan Boneau
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. J. C. Boudreaux
Center for Manufacturing
Engineering
National Bureau of Standards
Gaithersburg, MD 20899

Dr. Lyle E. Bourne, Jr.
Department of Psychology
Box 345
University of Colorado
Boulder, CO 80309

Dr. Gary L. Bradshaw
Psychology Department
Campus Box 345
University of Colorado
Boulder, CO 80309

kDr. Bruch Buchanan
Computer Science Department
University of Pittsburgh
222 Alumni Hall
Pittsburgh, PA 15260

LT COL Hugh Burns
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Mark Burstein
BBN
10 Moulton Street
Cambridge, MA 02138

Dr. Robert Calfee
School of Education
Stanford University
Stanford, CA 94305

Dr. Robert L. Campbell
IBM T.J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Dr. Joseph C. Campione
Center for th. Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61820

Dr. Jaime G. Carbonell
Computer Science Department
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Thomas Carolan
Institute for Simulation and Training
University of Central Florida
12424 Research Parkway
Suite 300
Orlando, FL 32826

Dr. Gail Carpenter
Center for Adaptive Systems
111 Cummington St., Room 244
Boston University
Boston, MA 02215

Dr. John M. Carroll
IBM Watson Research Center
User Interface Institute
P.O. Box 704
Yorktown Heights, NY 10598

CDR Robert Carter
Office of the Chief
of Naval Operations
OP-933D4
Washington, DC 20350-2000

Dr. Fred Chang
Pacific Bell
2600 Camino Ramon
Room 3S-450
San Ramon, CA 94583

Dr. Davida Charney
English Department
Penn State University
University Park, PA 16802

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Susan Chipman
Personnel and Training Research
Office of Naval Research
Code 1142CS
Arlington, VA 22217-5000

Dr. William J. Clancey
IRL
2550 Hanover Street
Palo Alto, CA 94304

Dr. Norman Cliff
Department of Psychology
Univ. of So. California
Los Angeles, CA 90089-1061

Dr. Paul Cohen
Computer Science Department
University of Massachusetts
Lederle Graduate Research Center
Amherst, MA 01003

Dr. Alan Collins
BBN
33 Moulton Street
Cambridge, MA 02238

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Greg Cooper
Stanford University
Knowledge Systems Lab
P. O. Box 8070
Stanford, CA 94305

Dr. Richard L. Coulson
Dept. of Physiology
School of Medicine
Southern Illinois University
Carbondale, IL 62901

Dr. Meredith P. Crawford
3563 Hamlet Place
Chevy Chase, MD 20815

Dr. Hans F. Crombag
Faculty of Law
University of Limburg
P.O. Box 616
Maastricht
The NETHERLANDS 6200 MD

Dr. Kenneth B. Cross
Anacapa Sciences, Inc.
P.O. Drawer Q
Santa Barbara, CA 93102

Dr. Cary Csichon
Intelligent Instructional Systems
Texas Instruments AI Lab
P.O. Box 660246
Dallas, TX 75266

Prof. Veronica Dahl
Department of Computer Science
Simon Fraser University
Burnaby, British Columbia
CANADA V5A 1S6

Dr. John F. Dalphin
Chair, Computer Science Dept.
Towson State University
Baltimore, MD 21204

Dr. Charles E. Davis
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217

Dr. Gerald F. DeJong
Dept. of Computer Science
University of Illinois
405 N. Mathews Ave.
Urbana, IL 61801

Dr. Thomas E. DeZern
Project Engineer, AI
General Dynamics
PO Box 748/Mail Zone 2646
Fort Worth, TX 76101

Dr. Thomas G. Dietterich
Dept. of Computer Science
Oregon State University
Corvallis, OR 97331

Dr. Ronna Dillon
Department of Guidance and
Educational Psychology
Southern Illinois University
Carbondale, IL 62901

Dr. J. Stuart Donn
Faculty of Education
University of British Columbia
2125 Main Mall
Vancouver, BC
CANADA V6T 1Z5

Dr. Kejitan Dontas
George Mason University
Dept. of Computer Science
4400 University Drive
Fairfax, VA 22030

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Pierre Duguet
Organization for Economic
Cooperation and Development
2, rue Andre-Pascal
75016 PARIS
FRANCE

Dr. Ralph Dusek
V-P Human Factors
JIL Systems
1225 Jefferson Davis Hwy.
Suite 1209
Arlington, VA 22201

Prof. Michael G. Dyer
Computer Science Department
UCLA
3532 Boelter Hall
Los Angeles, CA 90024

Dr. John Ellis
Navy Personnel R&D Center
Code 51
San Diego, CA 92252

Dr. Susan Epstein
144 S. Mountain Avenue
Montclair, NJ 07042

ERIC Facility-Acquisitions
4350 East-West Hwy., Suite 1100
Bethesda, MD 20814-4475

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Campus Box 345
Boulder, CO 80309-0345

Dr. Lee Erman
Technowledge, Inc.
525 University Avenue
Palo Alto, CA 94301

Dr. Tom Eskridge
Lockheed Austin Division
6800 Burleson Road
Dept. T4-41, Bldg. 30F
Austin, TX 78744

Dr. Lorraine D. Eyde
Office of Personnel Management
Office of Examination Development
1900 E St., NW
Washington, DC 20415

LCDR Micheline Y. Eyraud
Code 602
Naval Air Development Center
Warminster, PA 18974-5000

Prof. Lawrence M. Fagan
Stanford University Medical Center
TC-136
Medical Computer Science
Stanford, CA 94305

Dr. Brian Falkenhainer
Xerox PARC
3333 Coyote Hill Rd.
Palo Alto, CA 94304

Dr. Jean-Claude Falmagne
Department of Psychology
New York University
6 Washington Place
New York, NY 10003

Dr. Marshall J. Farr, Consultant
Cognitive & Instructional Sciences
2520 North Vernon Street
Arlington, VA 22207

Dr. P-A. Federico
Code 51
NPRDC
San Diego, CA 92152-6800

Dr. Jerome A. Feldman
International Computer
Science Institute
1947 Center Street
Berkeley, CA 94704-1105

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 19230
Springfield, IL 62708

Dr. Richard Fikes
Price Waterhouse Tech Center
68 Willow Road
Menlo Park, CA 94025

CAPT J. Finelli
Commandant (G-PTE)
U.S. Coast Guard
2100 Second St., S.W.
Washington, DC 20593

Dr. Douglas Fisher
Dept. of Computer Science
Box 67, Station B
Vanderbilt University
Nashville, TN 37235

Dr. Donald Fitzgerald
University of New England
Department of Psychology
Armidale, New South Wales 2351
AUSTRALIA

Mr. Nicholas S. Flann
Dept. of Computer Science
Oregon State University
Corvallis, Oregon 97331-3902

Dr. Kenneth D. Forbus
Department of Computer Science
University of Illinois
405 N. Mathews Avenue
Urbana, IL 61801

Dr. Kenneth M. Ford
Division of Computer Science
The University of West Florida
11000 University Parkway
Pensacola, FL 32514

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Charles Forgy
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. Mark Fox
Carnegie Mellon University
Robotics Institute
Pittsburgh, PA 15213

Dr. Carl H. Frederiksen
Dept. of Educational Psychology
McGill University
3700 McTavish Street
Montreal, Quebec
CANADA H3A 1Y2

Dr. John R. Frederiksen
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Dr. Norman Frederiksen
Educational Testing Service
(05-R)
Princeton, NJ 08541

Dr. Alfred R. Fregly
AFOSR/NL, Bldg. 410
Bolling AFB, DC 20332-6448

Dr. Peter Friedland
Chief, AI Research Branch
Mail Stop 244-17
NASA Ames Research Center
Moffett Field, CA 94035

Dr. Michael Friendly
Psychology Department
York University
Toronto ONT
CANADA M2J 1P3

Col. Dr. Ernst Frise
Heerespsychologischer Dienst
Maria Theresien-Kaserne
1130 Wien
AUSTRIA

Dr. Robert M. Gagne
1456 Mitchell Avenue
Tallahassee, FL 32303

Dr. Brian R. Gaines
Knowledge Science Institute
University of Calgary
Calgary, Alberta
CANADA T2N 1N4

Dr. Dedre Gentner
Department of Psychology
University of Illinois
603 E. Daniel
Champaign, IL 61820

Dr. Donald R. Gentner
Philips Laboratories
345 Scarborough Road
Briarcliff Manor, NY 10510

Dr. Helen Gigley
National Science Foundation
1800 G Street N.W.
Room 304
Washington, DC 20550

Dr. Philip Gillis
Army Research Institute
PERI-II
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Allen Ginsberg
AT&T Bell Laboratories
Holmdel, NJ 07733

Mr. Lee Gladwin
305 Davis Avenue
Leesburg, VA 22075

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Marvin D. Glock
101 Homestead Terrace
Ithaca, NY 14858

Dr. Dwight J. Goehring
ARI Field Unit
P.O. Box 5787
Presidio of Monterey
CA 93944-5011

Dr. Joseph Goguen
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Mr. Richard Golden
Psychology Department
Stanford University
Stanford, CA 94305

Mr. Harold Goldstein
University of DC
Department Civil Engineering
Bldg. 42, Room 112
4200 Connecticut Avenue, N.W.
Washington, DC 20008

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of
Technology
School of Industrial
and Systems Engineering
Atlanta, GA 30332-0205

Dr. Art Graesser
Dept. of Psychology
Memphis State University
Memphis, TN 38152

Dr. Wayne Gray
Artificial Intelligence Laboratory
NYNEX
500 Westchester Avenue
White Plains, NY 10604

H. William Greenup
Dep Asst C/S, Instructional
Management (E03A)
Education Center, MCCDC
Quantico, VA 22134-5050

Dr. Dik Gregory
Admiralty Research
Establishment/AXB
Queens Road
Teddington
Middlesex, ENGLAND TW110LN

Dr. Gleen Griffin
Naval Education and Training Program
Management Support Activity
Instructional Technology Impl. Div.
Code 0473
Pensacola, FL 32509-5000

Dr. Benjamin N. Grosof
IBM T.J. Watson Labs
P.O. Box 704
Yorktown Heights, NY 10598

Dr. Stephen Grossberg
Center for Adaptive Systems
Room 244
111 Cummington Street
Boston University
Boston, MA 02215

Michael Haben
DORNIER GMBH
P.O. Box 1420
D-7990 Friedrichshafen 1
WEST GERMANY

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. H. Hamburger
Department of Computer Science
George Mason University
Fairfax, VA 22030

Dr. Cheryl Hamel
Naval Training Systems Center
Code 712
12350 Research Parkway
Orlando, FL 32826

Dr. Bruce W. Hamill
Research Center
The Johns Hopkins University
Applied Physics Laboratory
Johns Hopkins Road
Laurel, MD 20707

Dr. Chris Hammond
Dept. of Computer Science
University of Chicago
1100 E. 58th Street
Chicago, IL 60637

Dr. Patrick R. Harrison
Computer Science Department
U.S. Naval Academy
Annapolis, MD 21402-5002

Dr. Peter Hart
301 Arbor Road
Menlo Park, CA 94025

Dr. Wayne Harvey
Center for Learning Technology
Education Development Center
55 Chapel Street
Newton, MA 02160

Dr. David Haussler
402 Nobel Drive
Santa Cruz, CA 95060

Dr. Barbara Hayes-Roth
Knowledge Systems Laboratory
Stanford University
701 Welch Road
Palo Alto, CA 94304

Dr. Frederick Hayes-Roth
Teknowledge
P.O. Box 10119
1850 Embarcadero Rd.
Palo Alto, CA 94303

Dr. James Hendler
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Dr. James Hiebert
Department of Educational
Development
University of Delaware
Newark, DE 19716

Dr. Geoffrey Hinton
Computer Science Department
University of Toronto
Sandford Fleming Building
10 King's College Road
Toronto, Ontario
CANADA M5S 1A4

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Haym Hirsh
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. James E. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

Dr. John H. Holland
Dept. of EE and CS
Room 3116
University of Michigan
Ann Arbor, MI 48109

Ms. Julia S. Hough
110 W. Harvey Street
Philadelphia, PA 19144

Dr. Jack Hunter
2122 Coolidge Street
Lansing, MI 48906

Dr. Ed Hutchins
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Wayne Iba
Dept. of Information and CS
University of California, Irvine
Irvine, CA 92717

Dr. Robin Jeffries
Hewlett-Packard Laboratories, 3L
P.O. Box 10490
Palo Alto, CA 94303-0971

Dr. Lewis Johnson
USC
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina Del Rey, CA 90292

Dr. Daniel B. Jones
U.S. Nuclear Regulatory
Commission
NRR/ILRB
Washington, DC 20555

Mr. Paul L. Jones
Research Division
Chief of Naval Technical Training
Building East-1
Naval Air Station Memphis
Millington, TN 38054-5056

Dr. Randolph Jones
Information and Computer Science
University of California
Irvine, CA 92717

Mr. Roland Jones
Mitre Corp., K-203
Burlington Road
Bedford, MA 01730

Prf. Aravind K. Joshi
Department of Computer Science
University of Pennsylvania
R-268 Moore School
Philadelphia, PA 19104

Dr. Gary Kahn
1220 Macon Avenue
Pittsburgh, PA 15218

Dr. Ruth Kanfer
University of Minnesota
Department of Psychology
Elliott Hall
75 E. River Road
Minneapolis, MN 55455

Dr. Michael Kaplan
Office of Basic Research
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Mr. Shyam Kapur
Dept. of Computer Science
Cornell University
4130 Upson Hall
Ithaca, NY 14853

Dr. Demetrios Karis
GTE Labs, MS 61
40 Sylvan Road
Waltham, MA 02254

Dr. A. Karmiloff-Smith
MRC-CDU
17 Gordon Street
London
ENGLAND WC1H OAH

Dr. Milton S. Katz
European Science Coordination
Office
U.S. Army Research Institute
Box 65
FPO New York 09510-1500

Dr. Smadar T. Kedar-Cabelli
NASA Ames Research Center
Mail Stop 244
Moffett Field, CA 94035

Dr. Frank Keil
Department of Psychology
228 Uris Hall
Cornell University
Ithaca, NY 14850

Dr. Richard M. Keller
Knowledge Systems Laboratory
Stanford University
Computer Science Dept.
Stanford, CA 94305

Dr. Wendy Kellogg
IBM T. J. Watson Research Ctr.
P.O. Box 704
Yorktown Heights, NY 10598

Dr. Douglas Kelly
University of North Carolina
Department of Statistics
Chapel Hill, NC 27514

Dr. J.A.S. Kelso
Center for Complex Systems
Building MT 9
Florida Atlantic University
Boca Raton, FL 33431

Prof. Larry Kerschberg
Dept. of Information System
& Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Dennis Kibler
Dept. of Information & Computer Science
University of California
Irvine, CA 92717

Dr. David Kieras
Technical Communication Program
TIDAL Bldg. 2360 Bonisteel Blvd.
University of Michigan
Ann Arbor, MI 48109

Dr. Thomas Killion
AFHRL/OT
Williams AFB, AZ 85240-6457

Dr. J. Peter Kincaid
Army Research Institute
Orlando Field Unit
c/o PM TRADE-E
Orlando, FL 32813

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Boulder, CO 80309-0345

Dr. Yves Kodratoff
George Mason University
AI Center
Fairfax, VA 22030-4444

Dr. Janet Kolodner
School of Information and
Computer Science
Georgia Institute of Technology
Atlanta, GA 30332-0280

Dr. Stephen Kosslyn
Harvard University
1236 William James Hall
33 Kirkland St.
Cambridge, MA 02138

Dr. Gary Kress
628 Spanier Avenue
Pacific Grove, CA 93950

Prof. Casimir A. Kulikowski
Department of Computer Science
Hill Center
for the Mathematical Sciences
Busch Campus
Rutgers University
New Brunswick, NJ 08903

Dr. David R. Lambert
Naval Ocean Systems Center
Code 772
271 Catalina Boulevard
San Diego, CA 92152-5000

Dr. Pat Langley
NASA Ames Research Center
Mail Stop 244-17
Moffett Field, CA 94035

Dr. Robert W. Lawler
Matthews 118
Purdue University
West Lafayette, IN 47907

Dr. Yuh-Jeng Lee
Department of Computer Science
Code 52
Naval Postgraduate School
Monterey, CA 93943

Ms. Debbie Leishman
Knowledge Scienc Institute
University of Calgary
Calgray, Alberta
CANADA T2N 1N4

Dr. Douglas B. Lenat
MCC
9430 Research Blvd.
Echelon Building #3
Austin, TX 78759

Dr. Alan M. Lesgold
Learning R and D Center
3939 O'Hara Street
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Keith R. Levi
Honeywell S and RC
3660 Technology Drive
Minneapolis, MN 55418

Dr. John Levine
Learning R and D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Leon S. Levy
2D Dorado Drive
Morristown, NJ 07960

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Matt Lewis
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Doris K. Lidtke
Software Productivity Consortium
1880 Campus Commons Drive, North
Reston, VA 22091

Dr. Sridhar Mahadevan
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Vern M. Malec
NPRDC, Code 14
San Diego, CA 92152-6800

Dr. Jane Malin
Mail Code EF5
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy
Code 04
NETPMSA
Pensacola, FL 32509-5000

Dr. Michel Manago
IntelliSoft
28 rue Georges Clemenceau
91400, ORSAY, FRANCE

Dr. William Mark
Lockheed AI Center
Building 259
2710 Sand Hill Rd.
Department 9006
Menlo Park, CA 94025

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Mr. John Mayer
University of Michigan
702 Church Street
Ann Arbor, Mi 48104

Dr. John McDermott
DEC
DTB5-3/E2
290 Donald Lynch Blvd.
Marlboro, MA 01752

Prof. David D. McDonald
Department of Computer
& Information Sciences
University of Massachusetts
Amherst, MA 01003

Dr. Joseph C. McLachlan
Code 52
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara Means
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Douglas L. Medin
Psychology Department
Perry Building
University of Michigan
330 Packard Rd.
Ann Arbor, MI 48104

Dr. Jose Mestre
Department of Physics
Hasbrouck Laboratory
University of Massachusetts
Amherst, MA 01003

Dr. Theodore Metsler
Department of the Navy
Office of the Chief of Naval Research
Arlington, VA 22217-5000

Dr. Alan L. Meyrowitz
Office of Naval Research, Code 1433
800 N. Quincy Rd.
Arlington, VA 22217

Dr. Ryszard S. Michalski
Department of Computer Science
George Mason University
4400 University Drive
Fairfax, Va 22030

Dr. Donald Michie
The Turing Institute
George House
36 North Hanover Street
Glasgow G1 2AD
UNITED KINGDOM

Dr. Vittorio Midoro
CNR-Istituto Tecnologie Didattiche
Via All'Opera Pia 11
GENOVA-ITALIA 16145

Dr. James R. Miller
MCC
3500 W. Balcones Center Dr.
Austin, TX 78759

Prof. Perry L. Miller
Dept. of Anesthesiology
Yale University School
of Medicine
333 Cedar Street
P. O. Box 3333
New Haven, CT 06510

Dr. Christine M. Mitchell
School of Indus. and Sys. Eng.
Center for Man-Machine
Systems Research
Georgia Institute of Technology
Atlanta, GA 30332-0205

Dr. Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Sanjay Mittal
Knowledge System Area
Intelligent Systems Lab
Xerox Palo Alto Research Center
Palo Alto, CA 94304

Dr. Andrew R. Molnar
Applic. of Advanced Technology
Science and Engr. Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
Naval Personnel R and D Center
San Diego, CA 92152-6800

Dr. Melvin D. Montemerlo
NASA Headquarters
Code RC
Washington, DC 20546

Dr. Raymond Mooney
Dept. of Computer Sciences
The University of Texas at Austin
Taylor Hall 2.124
Austin, TX 78712

Dr. Katharina Morik
GMD
F3/XPS
P.O. Box 1240
D-5205 St. Augustin
WEST GERMANY

Prof. John Morton
MRC Cognitive
Development Unit
17 Gordon Street
London WC1H OAH
UNITED KINGDOM

Dr. Jack Mostow
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. Randy Mumaw
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Dr. Kenneth S. Murray
Dept of Computer Sciences
University of Texas at Austin
Taylor Hall 2.124
Austin, TX 78712

Dr. William R. Murray
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue
Box 580
Santa Clara, CA 95052

Prof. Makoto Nagao
Dept. of Electrical Engineering
Kyoto University
Yoshida-Honmachi
Sakyo-Ku
Kyoto
JAPAN

Mr. J. Nelissen
Twente University of Technology
Fac. Bibl. Toegepaste Onderwyskunde
P. O. Box 217
7500 AE Enschede
The NETHERLANDS

Dr. T. Niblett
The Turing Institute
George House
36 North Hanover Street
Glasgow G1 2AD
UNITED KINGDOM

Dr. Ephraim Nissan
Department of Mathematics
& Computer Science
New Campus
Ben Gurion University of Negev
P. O. Box 653
84105 Beer-Sheva
ISRAEL

Dr. A. F. Norcio
Code 5530
Naval Research Laboratory
Washington, DC 20375-5000

Dr. Donald A. Norman
C-015
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Dr. Paul O'Rorke
Department of Information and
Computer Science
University of California
Irvine, CA 92717

Dr. Stellan Ohlsson
Learning R and D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. James B. Olsen
WICAT Systems
1875 South State Street
Orem, UT 84058

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Judith Reitman Olson
Graduate School of Business
University of Michigan
701 Tappan
Ann Arbor, MI 48109-1234

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Dr. Judith Orasanu
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Jesse Orlansky
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Prof. Tim O'Shea
Institute of Educational Technology
The Open University
Walton Hall
Milton Keynes MK7 6AA
Buckinghamshire, U.K.

Dr. Everett Palmer
Mail Stop 239-3
NASA-Ames Research Center
Moffett Field, CA 94035

Dr. Okchoon Park
Army Research Institute
PERI-2
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Ramesh Patil
MIT
Laboratory for Computer Science
Room 418
545 Technology Square
Cambridge, MA 02139

Dr. Michael J. Passani
Department of Computer and
Information Science
University of California
Irvine, CA 92717

Dr. Roy Pea
Institute for Research
on Learning
2550 Hanover Street
Palo Alto, CA 94304

Dr. Ray S. Perez
ARI (PERI-II)
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. C. Perrino, Chair
Dept. of Psychology
Morgan State University
Cold Spring La.-Hillen Rd.
Baltimore, MD 21239

Admiral Piper
PM TRADE
ATTN: AMCPM-TNO-ET
12350 Research Parkway
Orlando, FL 32826

Dr. Peter Pirolli
Graduate School of Education
EMST Division
4533 Tolman Hall
University of California, Berkeley
Berkeley, CA 94702

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Tomaso Poggio
Massachusetts Institute
of Technology E25-201
Center for Biological
Information Processing
Cambridge, MA 02139

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309-0345

Dr. Bruce Porter
Computer Science Department
University of Texas
Taylor Hall 2.124
Austin, TX 78712-1188

Mr. Armand E. Prieditis
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

Dr. J. Ross Quinlan
School of Computing Sciences
N.S.W. Institute of Technology
Broadway
N.S.W. AUSTRALIA 2007

Dr. Shankar A. Rajamoney
Computer Science Department
University of Southern California
Los Angeles, CA 91030

Mr. Paul S. Rau
Code U-22
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20903

Ms. Margaret Recker
Graduate Group in Education
EMST Division
4533 Tolman Hall
University of California, Berkeley
Berkeley, CA 94702

Dr. Stephen Reder
NWREL
101 SW Main, Suite 500
Portland, OR 97204

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Brian Reiser
Cognitive Science Lab
Princeton University
221 Nassau Street
Princeton, NJ 08544

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. J. Jeffrey Richardson
Center for Applied AI
College of Business
University of Colorado
Boulder, CO 80309-0419

Prof. Christopher K. Riesbeck
Department of Computer Science
Yale University
P. O. Box 2158, Yale Station
New Haven, CT 06520-2158

Prof. David C. Rine
Deptment of Computer
& Information Sciences
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Edwina L. Rissland
Dept. of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

LT CDR Michael N. Rodgers
Canadian Forces Personnel
Applied Research Unit
4900 Yonge Street, Suite 600
Willowdale, Ontario M2N 6B7
CANADA

Dr. Paul S. Rosenbloom
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina Del Ray, CA 90292

Dr. Ernst Z. Rothkopf
AT&T Bell Laboratories
Room 2D-456
600 Mountain Avenue
Murray Hill, NJ 07974

Dr. Allen A. Rovick
Rush Medical College
1653 W. Congress Parkway
Chicago, IL 60612-3864

Dr. Stuart J. Russell
Computer Science Division
University of California
Berkeley, CA 94720

Dr. Roger C. Schank
Northwestern University
Inst. for the Learning Sciences
1890 Maple
Evanston, IL 60208

Lowell Schoer
Psychological & Quantitative
Foundations
College of Education
University of Iowa
Iowa City, IA 52242

Dr. Jeffrey C. Schlimmer
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Janet W. Schofield
816 LRDC Building
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Paul D. Scott
University of Essex
Dept. of Computer Science
Wivenhoe Park
Colchester C043SQ
ENGLAND

Dr. Alberto Segre
Cornell University
Computer Science Department
Upson Hall
Ithaca, NY 14853-7501

Dr. Colleen M. Seifert
Dept. of Psychology
University of Michigan
350 Packard Rd.
Ann Arbor, MI 48104

Dr. Oliver G. Selfridge
GTE Labs
Waltham, MA 02254

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Michael G. Shafto
NASA Ames Research Ctr.
Mail Stop 239-1
Moffett Field, CA 94035

Dr. Valerie L. Shalin
Honeywell S&RC
3660 Technology Drive
Minneapolis, MN 55418

Dr. Jude W. Shavlik
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Mr. Colin Sheppard
AXC2 Block 3
Admiralty Research Establishment
Ministry of Defence Portsdown
Portsmouth Hants P064AA
UNITED KINGDOM

Dr. Ben Shneiderman
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Dr. Jeff Shrager
Xerox PARC
3333 Coyote Hill Rd.
Palo Alto, CA 94304

Dr. Howard Shrobe
Symbolics, Inc.
Eleven Cambridge Center
Cambridge, MA 02142

Dr. Randall Shumaker
Naval Research Laboratory
Code 5510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Bernard Siler
Information Sciences
Fundamental Research Laboratory
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Herbert A. Simon
Departments of Computer Science and
Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Robert L. Simpson, Jr.
DARPA/ISTO
1400 Wilson Blvd.
Arlington, VA 22209-2308

Dr. Zita M. Simutis
Chief, Technologies for Skill
Acquisition and Retention
ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Derek Sleeman
Computing Science Department
The University
Aberdeen AB9 2PX
Scotland
UNITED KINGDOM

Ms. Gail K. Slemon
LOGICON, Inc.
P.O. Box 85158
San Diego, CA 92138-5158

Dr. Edward E. Smith
Department of Psychology
University of Michigan
330 Packard Road
Ann Arbor, MI 48103

Dr. Reid G. Smith
Schlumberger Technologies Lab.
Schlumberger Palo Alto Research
3340 Hillview Avenue
Palo Alto, CA 94304

Dr. Elliot Soloway
EE/CS Department
University of Michigan
Ann Arbor, MI 48109-2122

Linda B. Sorisio
IBM-Los Angeles Scientific Center
11601 Wilshire Blvd., 4th Floor
Los Angeles, CA 90025

Dr. N. S. Sridharan
FMC Corporation
Box 580
1205 Coleman Avenue
Santa Clara, CA 95052

Dr. Frederick Steinheiser
CIA-ORD
Ames Building
Washington, DC 20505

Dr. Ted Steinke
Dept. of Geography
University of South Carolina
Columbia, SC 29208

Dr. Leon Sterling
Dept. of Computer Engineering
and Science
Crawford Hall
Case Western Reserve University
Cleveland, Ohio 44106

Dr. Michael J. Strait
UMUC Graduate School
College Park, MD 20742

Dr. Devika Subramanian
Department of Computer Science
Cornell University
Ithaca, NY 14853

Dr. Patrick Suppes
Stanford University
Institute for Mathematical
Studies in the Social Sciences
Stanford, CA 94305-4115

Dr. Richard Sutton
GTE Labs
Waltham, MA 02254

Dr. William Swartout
USC
Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292

Mr. Prasad Tadepalli
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. Gheorghe Tecuci
Research Institute for Computers
and Informatics
71316, Bd. Miciurin 8-10
Bucharest 1
ROMANIA

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Chris Tong
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. Douglas Towne
Behavioral Technology Labs
University of Southern California
1845 S. Elena Ave.
Redondo Beach, CA 90277

Lt. Col. Edward Trautman
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826

Dr. Paul T. Twohig
Army Research Institute
ATTN: PERI-RL
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Paul E. Utgoff
Department of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Dr. Harold P. Van Cott
Committee on Human Factors
National Academy of Sciences
2101 Constitution Avenue
Washington, DC 20418

Dr. Kurt Van Lehn
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. W. S. Vaughan
800 N. Quincy Street
Arlington, VA 22217

Dr. Adrian Walker
IBM
P. O. Box 704
Yorktown Heights, NY 10598

Dr. Diana Wearne
Department of Educational
Development
University of Delaware
Newark, DE 19711

Prof. Sholom M. Weiss
Department of Computer Science
Hill Center for Mathematical
Sciences
Rutgers University
New Brunswick, NY 08903

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1205 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. David C. Wilkins
Dept. of Computer Science
University of Illinois
405 N. Mathews Avenue
Urbana, IL 61801

Dr. Kent E. Williams
Institute for Simulation and Training
The University of Central Florida
12424 Research Parkway, Suite 300
Orlando, FL 32826

Dr. Marsha R. Williams
Applic. of Advanced Technologies
National Science Foundation
SEE/MDRISE
1800 G Street, N.W., Room 635-A
Washington, DC 20550

S. H. Wilson
Code 5505
Naval Research Laboratory
Washington, DC 20375-5000

Dr. Patrick H. Winston
MIT Artificial Intelligence Lab.
545 Technology Square
Cambridge, MA 02139

Dr. Edward Wisniewski
Honeywell S and RC
3660 Technology Drive
Minneapolis, MN 55418

ONR DISTRIBUTION LIST [ILLINOIS/WILKINS]

Dr. Paul T. Wohig
Army Research Institute
5001 Eisenhower Ave.
ATTN: PERI-RL
Alexandria, VA 22333-5600

Dr. Joseph Wohl
Alphatech, Inc.
2 Burlington Executive Center
111 Middlesex Turnpike
Burlington, MA 01803

Dr. Beverly P. Woolf
Dept. of Computer and
Information Sciences
University of Massachusetts
Amherst, MA 01003

Dr. Ronald R. Yager
Machine Intelligence Institute
Iona College
New Rochelle, NY 10801

Dr. Masoud Yasdani
Dept. of Computer Science
University of Exeter
Prince of Wales Road
Exeter EX44PT
ENGLAND

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550

Dr. Maria Zemankova
National Science Foundation
1800 G Street N.W.
Washington, DC 20550

Dr. Uri Zernik
GE - CRD
P. O. Box 8
Schenectady, NY 12301