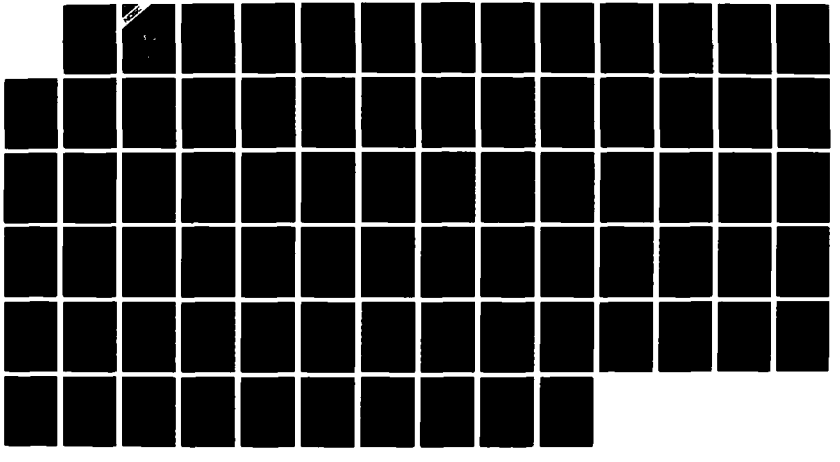
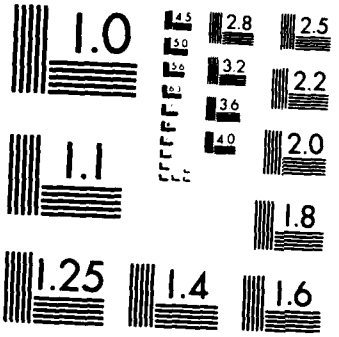
AD-A188 597 KNOWLEDGE ACQUISITION: A REVIEW OF TOOLS AND IDEAS(U) 1/1
NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA R B KAMEN
AUG 87 NOSC/TR-1095

UNCLASSIFIED F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

DTIC FILE COPY

NOSC TR 1095

NAVAL OCEAN SYSTEMS CENTER San Diego, California 92152-5000

(4)

NOSC TR 1095

Technical Report 1095
August 1987

Knowledge Acquisition: A Review of Tools and Ideas

R. B. Kamen

AD-A188 597

DTIC
ELECTE
DEC 2 3 1987
S D
CAD

Approved for public release; distribution is unlimited.

87 12 14 070

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

The work described in this report was sponsored by the Office of Naval Research as part of program element 62721N, task number RF21244, work unit number 440-CC96. The LPS accession number is DN188 532.

Glen R. Allgaier
Head, Artificial Intelligence Branch

Willis T. Rasmussen
Head, Advanced C^2 Technologies Division

ACKNOWLEDGEMENT

The author wishes to thank Russell Farris for his assistance in organizing and reviewing this report.

PK

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD-A188 597

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) TR 1095	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b OFFICE SYMBOL (if applicable) Code 444	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b ADDRESS (City, State and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research 8b OFFICE SYMBOL (if applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State and ZIP Code) Arlington, VA 22217	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 62721N; PROJECT NO:; TASK NO: RF21244; AGENCY ACCESSION NO: DN188 532

11 TITLE (Include Security Classification)
Knowledge Acquisition: A Review of Tools and Ideas

12 PERSONAL AUTHOR(S)
R.B. Kamen

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM Oct 84 TO Oct 85	August 1987	79

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GROUP	Knowledge acquisition, Knowledge-based systems, Expert systems, Air strike planning, Artificial intelligence,

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

The principal bottleneck in the development of expert systems is the acquisition and refinement of knowledge. This report describes a number of tools, languages, and ideas that were reviewed while looking for ways of acquiring knowledge for the Air Strike Planning Advisory (ASPA) system.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL R.B. Kamen	22b TELEPHONE (Include Area Code) (619) 225-6571 22c OFFICE SYMBOL Code 421

DD FORM 1473, 84 JAN

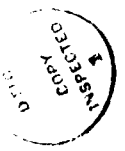
83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

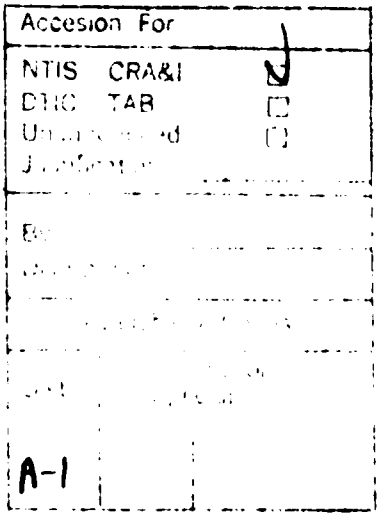
UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

CONTENTS (continued)

CONTENTS (continued)





SUMMARY

OBJECTIVE

Search for tools or ideas that could be used to aid in the development of the Air Strike Planning Advisor (ASPA) system. Review current literature.

RESULTS

Found many interesting and useful tools. However, none could be applied directly to solving the problem of acquiring knowledge for the ASPA.

RECOMMENDATIONS

Develop a tool based on constraint satisfaction problem solving. Draw upon work in progress at the University of California, Los Angeles (UCLA) which focuses on converting some constraint graphs to trees. Make these conversions the basis for generating optimal orderings for making decisions in constructing plans.

INTRODUCTION

We are living proof that knowledge acquisition is possible

PROBLEM

Expert systems, i.e., computer systems capable of performing tasks normally performed by human experts, are in great demand. Currently, the largest single problem in developing such systems is knowledge acquisition (KA). In addition to the time spent in computer programming, building an expert system may take months or years of work on the part of knowledge engineers who collect and organize the knowledge, and the human experts who provide the knowledge.

OBJECTIVE

The objective of the work described in this report was to search for tools or ideas that could be used to aid in the development of the Air Strike Planning Advisor (ASPA) system.

APPROACH

The approach was to search for ideas and techniques that could be used to facilitate KA. An effort was made to show how these ideas could be used to gather and refine knowledge for eventual use by expert systems.

Because this is a new technology, it will be necessary to refine, elaborate, and quantify many of the ideas covered here before they can be used. Tools and ideas that did not appear to have any direct or indirect application to ASPA have not been described here.

RELATED DOCUMENTS

NOSC Technical Report 1094, "Knowledge Acquisition Methodology," discusses the knowledge acquisition problems from the standpoint of the social sciences.

BACKGROUND

Because of the newness and complexity of the knowledge acquisition problem, the background of the knowledge acquisition problem is discussed in detail in the following two sections.

AN OVERVIEW OF EXPERT SYSTEMS

DIFFERENCES BETWEEN CONVENTIONAL AND EXPERT SYSTEMS

Flexibility. In conventional systems, fixed programs operate on expected data in known formats. In expert systems, the system is often data- or event-driven in the sense that patterns in the data control the selection of code to be activated. Conventional systems are flexible enough to handle more than one set of input data for a particular type of task, but they cannot handle more than one type of task.

Explicitness of knowledge. The performance of conventional systems is accomplished by knowledge which, although built into the system, is not explicitly stated. It is difficult to add new knowledge to such a system because alterations may be needed throughout to achieve the correct changes in how the system operates on the data. In contrast, knowledge-based expert systems generally contain explicit representations of the knowledge responsible for their performance. New knowledge can be added in one place with confidence that it will be correctly integrated into the functioning of the system. Several people can contribute knowledge to such a system. Thus, expert systems may be a repository of knowledge accumulated from specialists with diverse experience, so the system may ultimately attain a level of expertise exceeding that of any of its "tutors."

Self-improvement. In conventional systems, it is necessary to rewrite and recompile the system to get it to do something differently; in knowledge-based systems, the modularity of control structures, data, and heuristics for manipulating data make it possible to run the system with different sets of heuristics for comparison in an effort to evolve those heuristics into an improved set. Michalski & Chilausky (1980b) describe a system for diagnosing soybean diseases in which rules devised by a plant pathologist were compared with rules generated by machine on the basis of examples. In handling soybean data, consisting of 19 diseases with 35 descriptors (domain sizes from 2 to 7 characters) on 307 cases, the system, in its two different forms, was presented with a test set of 376 new cases. The version with the plant pathologists' original rules yielded an accuracy of 83 percent on the test set. The version with the machine-generated rules yielded an accuracy of better than 99 percent.

Man-machine interface. Conventional systems provide users with a single mode of usage in which the user is the client. However, expert systems allow the user to be a tutor, thus improving and extending the knowledge in the system, or a pupil. In this mode, the knowledge in the expert system can be harvested for human use by obtaining explanations as a result of challenging and examining the reasoning processes used in obtaining answers; the system may be used to train new human experts; and the system may be used to encourage the exposure and refinement of hitherto private heuristic knowledge. The kind of study, critique, and explicit teaching that takes place for other scientific knowledge is possible for the heuristics of a field as well.

ARCHITECTURE OF EXPERT SYSTEMS

The software for conventional systems is usually divided into two parts: program and data. The software for expert systems usually contains three parts: data, knowledge base, and control.

Data. The data are in data structures that are examined and modified by the system as progress is made in solving the problem. This component is analogous to short-term memory, even if some data remains unchanged for long periods of time. It is sometimes called the global database or blackboard because it is accessible to all parts of the program.

Knowledge Base. The knowledge base contains information about the domain and rules (heuristics) for manipulating the information, that is, the reasoning techniques used in the domain. The information and rules taken together constitute the domain knowledge. The rules include both the hard and fast rules (regulations, specifications, etc.), and rules of thumb (judgemental rules of the field and rules of good practice). To some extent, the knowledge may be experimental and uncertain. The rules are of the sort humans learn during internships, Ph.D. programs, and apprenticeships, or rules which they develop over years of experience in a field.

The rules are usually entered into the computer as "production rules." The simplest production rules have the form: IF such-and-such condition exists, THEN perform such-and-such. Production rules have been called operations, inference rules, antecedent-consequent pairs, condition-action pairs, production, situation-action pairs, and knowledge-based rules. When the data satisfies the antecedent condition, we say there is a pattern match. Rules do not "call" other rules as subroutines do in conventional programming. This part of the program is analogous to long-term memory. The whole knowledge-based system is sometimes correspondingly called a rule-based system, a production system, or a pattern-directed inference system.

Control. The control part of the expert system--sometimes called the inference engine--performs scheduling, selection, conflict resolution, and activation of the production rules. The amount of control structure may vary from virtually nothing to a complete algorithm. The control structure is sometimes called the inference engine. Both deductive and inductive inference can be simulated. Repeated applications of Bayes theorem to combine conditional probabilities of various events are used in some systems.

The control structure is generally independent of the knowledge base, thus, it is sometimes possible to use the control structure as a shell within which a new expert system can be developed.

SHELLS

Expert system shells are essentially the control part of an expert system without its data or knowledge base. Using shells facilitates the development of families of expert systems. EMYCIN was constructed using the control structure from MYCIN, an expert system for diagnosing and treating bacterial infections of the blood. The inference engine from EMYCIN was used with expert knowledge in the domain of pulmonary function diagnosis to create PUFF, a system for interpreting lung tests. It was also used to build SACON (for structural analysis) and CLOT (to study blood clotting).

The expert systems shell EXPERT was created by removing the data and knowledge base from CASNET, a system for medical diagnosis.

The shells just described are sometimes called domain-independent expert systems. All of them provide a single-knowledge representation and inference mechanism. Starting with one of these "empty" expert systems can accelerate the process of building an expert system, since some of the development work has already been done, and some of the decisions among alternatives for knowledge representation are made by the choice of an expert system shell.

ADVISORY SYSTEMS

Advisory systems are expert systems that are designed to give advice rather than to actually perform a task. Advisory systems can be used to perform the following tasks:

1. Simulate decision-making processes.
2. Suggest the n best approaches, or generate the n best plans.
3. Uncover issues and factors to weigh in making critical decisions associated with a task.
4. Assess the users' plans.
5. Critique the best known approach to encourage its evolution into a better one.
6. Elaborate on the implications of high-level decisions, especially where estimating the effects of these decisions requires significant amounts of computation.
7. Suggest alternate solutions or approaches, discussing the risks and benefits of each, and rating them by predetermined criteria.

The most advanced general purpose advisory system is a decision analyst--GODDESS--developed at UCLA.

AN OVERVIEW OF KNOWLEDGE ACQUISITION

USES OF KNOWLEDGE ACQUISITION (KA) TOOLS

The KA tools currently available have been designed to facilitate the process of KA, not to replace human knowledge engineers. In fact, for many years to come the primary users of KA tools will be the knowledge engineers, not the domain experts.

KA tools are used to perform these tasks:

1. Speed the acquisition of knowledge.
2. Improve the quality of knowledge.
3. Elicit knowledge which may be difficult to articulate.

4. Detect inconsistencies, conflicts, and ambiguities in knowledge.

5. Recognize gaps in knowledge.

6. Provide feedback to the expert problem solver to obtain clarifications in cases where the system's interpretation of newly acquired information may be at variance with the expert's interpretation.

7. Protect against inappropriate use of knowledge.

8. Protect against clerical and other errors in entering information into the machine.

KINDS OF KNOWLEDGE

There are many kinds of knowledge used in solving real-world problems. Appendix A contains a list of kinds of metaknowledge and Appendix B lists kinds of general and domain specific knowledge.

IDENTIFYING KNOWLEDGE FOR USE IN EXPERT SYSTEMS

At first glance, KA appears to be concerned only with developing techniques to assist the expert and knowledge engineer in entering knowledge into its machine representations to produce expert performance. However, this view is too narrow. The principal difficulty in knowledge acquisition is not in translating knowledge into machine processable forms, but in finding out what knowledge should be obtained and placed in the machine to produce the desired performance.

This is an especially irksome problem for knowledge engineers because the domain experts are seldom able to present their knowledge in an orderly, consistent, and complete way. Furthermore, it usually requires a great deal of effort to separate the relevant knowledge from the experts' other knowledge.

PROBLEMS OF KNOWLEDGE ACQUISITION

The phrase, "knowledge acquisition," has been used in a variety of ways to encompass different problems, including those listed here:

1. Given the expert and the problem, find out what knowledge possessed by the expert is used in handling the problem.

2. Given the expert and a domain of problem tasks, find out what knowledge the expert has to have to be able to handle any of those problem tasks.

3. Given pre-identified knowledge of the domain and the problem tasks, find representations of the knowledge which will support the solution of the problems in the machine.

4. Given pre-identified knowledge of the domain and pre-identified representations, find a technique for transferring the knowledge from one to the other.

5. Given pre-identified knowledge, an established transfer method, and representation, find a technique to assure the accuracy of the system's interpretation of the transferred knowledge.

EXPERTISE

The information and skills that make up the kinds of knowledge we refer to as expertise are difficult to express in discrete, concrete, and machine-usable terms. Difficulties arise in this process because domain experts typically employ "compiled" knowledge--shortcuts--in their own problem solving activities. The "compiled" knowledge tends to be harder to articulate and explain, particularly to someone not familiar with the domain. Less expert practitioners in a domain can often explain the principles behind their decisions more readily than the experts can. This may be because they do not have enough experience to have devised the shortcuts in thinking which are typical of the true experts.

These shortcuts, of course, are just what is needed to constrain search in problem solving, and techniques for knowledge acquisition must be oriented to capture knowledge of this form.

FORMULATING PROBLEMS

When building expert systems, decisions are needed on problem formulation and, in the context of those formulations, techniques for identifying knowledge, for acquiring it, for organizing it, for representing it, and for compiling it for efficient use by machine.

Devising a system for formulating problems can help system builders identify relevant knowledge. The need to accommodate shifts in problem formulation, which may be essential to performance efficiency of the completed system, may complicate the representation of the system's knowledge. Ideally, a knowledge acquisition tool would be able to request reformulations of a given problem, and be able to choose among them.

VOCABULARY

The knowledge engineer must begin the acquisition of knowledge for a new field as a novice. Unfortunately, the vocabulary used to discuss a problem with a novice is not always adequate for solving the problem (Buchanan et al., 1983). Some features of the domain may be abstractions for which the expert has no names. These features, nonetheless, need to be identified, articulated, and represented.

STEPS IN KNOWLEDGE ACQUISITION

The knowledge acquisition process typically includes the following steps:

1. Identify the knowledge available and the knowledge that will be required.

a. Identify the range and characteristics of problems the system will handle.

b. Identify primitive terms and concepts.

c. Identify strategies for handling the problems.

d. Identify data sources the domain expert draws from during problem solving.

e. Identify tasks and subtasks associated with the problems.

f. Identify which parts of the information will need to be reworked into forms that will make it usable.

2. Choose and implement formalisms for expressing knowledge.

   a. Design a structure for organizing the knowledge.

   b. Choose languages and tools.

   c. Choose knowledge representations for all the different types of knowledge encountered, taking care to match the representation with what is expressed and with the way it will be used in the system.

3. Transfer identified knowledge into formalisms.

4. Validate the KA process, i.e., determine whether the knowledge functions as expected in the system.

   a. Ensure the consistency of the acquired knowledge.

   b. Verify the knowledge to eliminate errors.

   c. Eliminate redundancies and redundant information to improve efficiency.

   d. Ensure that unanticipated interactions between parts of the system do not prevent it from using the knowledge it contains appropriately.

   e. Ensure that the knowledge is complete with respect to the tasks that the system must perform.

The steps given above must not be thought of as sequential steps since there is an enormous amount of feedback among them and the other tasks associated with developing the system. The standard process of expert system development is a repetitive one of generating and testing systems which grow incrementally from an initial minimal system to one containing the appropriate domain knowledge for adequate performance. Deficiencies revealed by early tests are clues that indicate what knowledge is missing from the initial versions which will need to be incorporated into later versions.

IDENTIFYING/LIMITING THE DOMAIN

The processes of defining the domain and acquiring knowledge are deeply intertwined, but the knowledge engineer should begin by trying to define the boundaries of the domain, and of the problems to be solved in the domain. The definitions will be modified as knowledge accumulates.

IDENTIFYING THE DOMAIN

The product of domain definition should include the following items:

1. A general description of the types of problems the system will be expected to handle.
2. References to written sources and where they exist.
3. An annotated collection of basic concepts, terms, acronyms, and symbols.
4. Identification of experts.
5. Examples of reasoning used in various scenarios.
6. Realistic expectations for system performance.
7. A core system in which all of the components of an operational system are functioning, but without the breadth or depth of a final system.
8. A catalog of domain entities (objects), relationships among objects, interactions among objects, and object descriptions, including abstract descriptions and generalizations over multiple objects and classes of objects. This will express the domain's ontology.
9. A definition of input formats and sources.
10. Information on how a given specific problem will be expresssed to the system when the system is used.
11. An initial-state description including "background" knowledge.
12. A fundamental set of reasoning and analysis rules including heuristic rules (rules of thumb).
13. Strategies (metarules) capable of functioning as part of the control knowledge for the system.

DOCUMENTING THE KA PROCESS

The KA work may take months or years and involve a large number of people. To prevent the KA team from working at cross purposes, and to enable new people to understand the intent and background of the work, it is suggested that documents describe in detail exactly what is known during different phases of the project (Grover, 1983). The resulting collection of documents may be used as a common medium for communication and reference, and even as a partial substitute for the expert(s) during the latter stages of the project.

METHODS OF ACQUIRING KNOWLEDGE

Methods of acquiring knowledge include interviewing, collecting plans, and metaplanning.

INTERVIEWING

Some of the techniques used to obtain knowledge from experts are the following:

- Protocol analysis
- Goal decomposition
- Scenario "walk through "
- Reclassification
- Expert's justification of his or her methods
- Limitations on the range of topics covered

Protocol analysis. Protocol analysis consists of having the expert attempt to solve an actual problem in a manner with which the expert is familiar, usually in the domain setting. The interviewer observes this process and interrupts periodically to obtain information on the expert's reasoning processes. Although these interruptions may be intrusive, they are necessary to ensure that the interviewer is eliciting rather than inferring the actual reasoning (Simon, 1980).

Goal decomposition. Goal decomposition is a process of reducing a big problem into a number of smaller ones. The decomposition process can yield insight into the domain and make it manageable, but this technique cannot be used alone since it will not provide the details needed for interpreting situations and objects in the domain.

Scenario walk throughs. The scenario walk through is a technique which requires the expert to verbally reconstruct his problem-solving behavior. Usually a very elementary scenario is chosen (by the expert) to illustrate the reasoning he uses in solving a problem. The scenario chosen for the walk through should be:

1. Well understood

2. Archetypal

3. Important

4. Minimal (does not incorporate any unnecessary complexities)

5. Expected (experts are not in disagreement over any important aspect) (Grover, 1983)

In the process of walking through the scenario, there are apt to be many distractions, including the appearance of terms and concepts unfamiliar to the interviewer, who must decide whether to interrupt the process for elaborations on the spot or to make note of gaps to be filled in later.

Walking through the scenario may be complicated by "rule fanout." The expert is considering many rules during the reasoning process, and is liable to get bogged down in explanations of all the aspects of reasoning and either never completes the scenario, or forgets to mention critical steps which, later, will prevent the system from functioning successfully.

Reclassification. Reclassification attempts to refine the classification of features of the domain into specific objects and activities. In an object-oriented problem domain, this approach extracts valuable knowledge of objects and enables the knowledge engineer to fill in the object hierarchies and describe their inheritance properties.

Expert's justification of his or her knowledge. Requests that the expert try to justify the knowledge can lead to the additional knowledge needed to build the system. The justifications start from already identified knowledge. They proceed by examining that knowledge from the viewpoint of explaining how the expert has confidence that the putative knowledge is correct, how the knowledge is known, and on what assumptions or principles that knowledge is based. Justifications may also be sought which will explain why knowing that element of the knowledge is adequate within its framework to handle the problem at some level. These techniques can elicit fundamental knowledge that is not consciously associated with any single task because it applies to so many tasks.

Limitations on the range of topics covered. During the performance of a task, only a small proportion of an expert's knowledge is applicable at one time. Thus, it makes sense to set up situations in which the focus is on acquiring selected portions of knowledge during specific interviews. The expert can often recommend appropriate situations. Once a limited area is chosen, an attempt can be made to draw out everything relevant to that situation, including pointers to other topics that need to be explored in subsequent interviews.

Ideally, focus of an interview should be limited to subareas selected during the decomposition phase.

Recommendations from a domain expert. In Snell (1983), a domain expert recommends that interviewers keep in mind the following points:

- Do not rush the expert.
- Expect ambiguity.
- Use existing documentation.
- Let the expert do the talking.
- Keep rules as self-evident as possible.
- Avoid talking about performance expectations of the system during the early phases.

INTERACTION BETWEEN KNOWLEDGE ENGINEERS AND EXPERTS

In the Dipmeter Advisor project, knowledge engineers found that the experts sometimes were "moving targets." An expert was using the process of program construction as a test bed for his own evolving ideas. This may be an undocumented future use of expert systems for training and discovery, but it is also an added complication.

Also, experts' reasoning processes were somewhat different depending upon whether the scenario was picked by themselves or chosen for them (Smith & Baker, 1983). The difference may have been a natural result of the greater expertise that they were able to muster on the problems of their choice. This should not be an argument against having the expert choose his own problems, since it is just this greater expertise which is of primary interest to the knowledge acquisition team. On the other hand, attempting to solve problems on the edge of his area of expertise may cause the expert to recall some of the principles that might not otherwise have been remembered.

COLLECTING PLANS OR PARTIAL PLANS

Collecting and indexing plans or partial plans may make sense for acquiring expertise which is to be used in systems that will construct or modify analogous plans. Along with collecting the plans, it will be necessary to identify their salient characteristics and the rules for generalizing, modifying, or making the analogies.

Plans collected early in the KA process can provide structures to guide in collecting and refining of data for analogous plans. Extracting the structures could be accomplished automatically in some cases by using KA tools that use inductive methods.

METAPLANNING

Metaplanning is a technique in which operations carried out during plan derivation are themselves regarded as actions to be planned at a higher level. The operations which are subject to metaplanning include constraint propagation, top-down elaboration, or backward and

forward extension. Knowledge acquirers should watch for signs of any metaplanning that the experts are doing.

CONFLICT RESOLUTION FOR SYSTEMS DRAWN FROM MULTIPLE EXPERTS

Where substantial conflict is expected from experts in a domain, it is advisable to choose conflict resolution strategies and set them in place as early as possible. Ideally, these strategies need to be decided upon before any conflicts actually arise. However, it may not be necessary to confront experts for a resolution of some kinds of conflict. Some apparent conflicts may be reflections of the grainsize of the variables. Also, apparent conflicts may be indications that the items in dispute have little effect on the solution of the problems being examined.

If there is doubt as to which expert is right, and if the system has sufficient modularity, several versions of the system may be developed and tested. An abstract space which spans over the apparently conflicting information may be maintained by entering a range of values as opposed to a single specific value for some concepts or terms. The version space which results will take advantage of the commonalities of different experts without commitment to areas of differences. Ultimately, it may be necessary to make the commitment but, in some cases, this approach will result in abstractions which are not only adequate to produce satisfactory performance, but also superior in terms of computational power and efficiency.

DECISION ANALYSIS

A formal decision method can be viewed as a closed inferential procedure whose input is an axiomatic description of a decision process and whose output is a prescribed action. Decision analysis is a collection of techniques for helping a decision maker develop insight with respect to his decision. These techniques assist in challenging the conceptual framework in which a decision problem is initially described, and they can help in reformulating problems so that they are more readily handled. Human beings are very good at identifying the factors that are relevant to the problem, but machines are generally unable to accomplish this task by formal techniques (McCarthy, 1980). But the situation is reversed when the task is to sort in order of importance the elements of a list of factors that are relevant to a decision. In tasks that require rank-ordering relevant factors, formal techniques tend to significantly outperform common sense. For the purposes of this discussion, a factor is counted as important on the basis of its ability to change a recommendation for action as its own values change.

Attention-focusing methods from decision analysis can be used as preprocessors to simplify the construction of complex models. This also makes it easier for nonexperts to comprehend the decision process, since part of the problem has been decomposed and can be addressed separately.

The object of modeling in the context of decision analysis is to devise a formal representation of a real-world decision problem. The analyst obtains a set of variables that are relevant to the decision problem and specifies the relationships among them (Matheson & Howard, 1977). The modeling effort produces an explicit value model to determine the relative desirability of various outcomes under consideration (Howard, 1973; Keeney & Raiffa, 1976).

Once a formal model of the decision problem has been obtained, the planning formalism infers a strategy (i.e., a plan) for action. The most widely accepted planning paradigm in decision analysis is to select the strategy that maximizes expected utility (Howard, 1968, 1973; Keeney & Raiffa, 1976; Matheson & Howard, 1977). However, this paradigm may not be general enough (March, 1978; Tversky & Kahneman, 1974, 1981; Sacerdoti, 1975).

Although most of the academic research in the field of decision analysis has been devoted to the development of planning techniques (Howard, 1973; Simon, 1977), newer work is addressing modeling techniques which are critical for the identification phase of knowledge acquisition (Holtzman, 1981, 1984).

The suitability of formal decision-making techniques for the evaluation of the importance of factors in a decision makes these techniques desirable for helping to focus attention on the appropriate concepts during the formalization and clarification phases of knowledge acquisition.

MACHINE LEARNING

Machines with the ability to learn show promise of acquiring expertise in specific domains. The machine's ability to learn will affect the types and structure of the data gathered for it.

Learning strategies. Learning strategies differ from each other in the type and amount of inference they must perform to derive the desired knowledge. Some systems perform complex inferences that occasionally lead to the discovery of new knowledge. The learning strategies used in expert systems include:

1. Learning from instruction, in which new knowledge is integrated with prior knowledge for effective retrieval and use. Versions of this strategy of learning are implemented in, for example, TEIRESIAS and NANOKLAUS.

2. Learning by deductive inference, in which the inference permits the system to determine important consequences of the knowledge, or to restructure the knowledge into more useful or effective forms. Automatic theorem provers use this learning strategy.

3. Learning by analogy, in which existing knowledge is transformed or extended to make it apply to similar tasks.

4. Learning from examples, in which examples are typically classified by a human teacher or another system. This inductive learning system forms or modifies decision rules by generalizing from the examples.

5. Learning by experimentation, in which the system finds examples by search or some other active effort. The system then learns from the examples. This method is used in the LEX symbolic integration learning system.

6. Learning by observation and discovery, in which the system develops theories about observed or given facts.

INDUCTIVE METHODS

The Knowledge Acquisition (KA) process can be simplified by having the machine use inductive techniques to learn rules directly from examples of decisions made by human experts.

One of the benefits of using an inductive form of knowledge acquisition is that it relieves the system builders of the task of formalizing the expert's knowledge.

Computer programs which induce rules from examples of expert decisions are a promising approach to rule acquisition. In some cases, rules formed by such programs have outperformed rules written by human experts (Quinlan, 1983; Michalski & Chilausky, 1980b). However, more foundational work is needed to provide explanation from system generated rules, and the question of validating these rules is felt to be more difficult (although it is not more difficult in principle) because the rules are sometimes too complex to be used or understood by humans. Inductively derived rules are more likely to be comprehensible if the domain or problem is decomposed ahead of time by the domain expert, but this process is itself very difficult and some domains may not be easily decomposed. A start on automating the decomposition of domain knowledge is presented in new work on conceptual clustering (Michalski & Stepp, 1983; Langley et al., 1986; Fisher & Langley, 1985; Nordhausen, 1986) and in some work on dividing examples into a hierarchy of subclasses (Paterson, 1983; Fisher, 1986).

Additional research is required to incorporate knowledge into the search for appropriate generalizations. The combinatorial explosion can best be controlled by using domain knowledge during the inductive process. Quinlan's ID3 algorithm, for example, uses an information theoretic measure to select the next attribute in its decision tree (Quinlan, 1979). Work on GEM, which is to be a part of the ADVISE system for building expert systems, is progressing in this area. GEM will permit the user to specify background domain knowledge in a stylized language.

In spite of their current limitations, inductive approaches to knowledge acquisition are the most promising. The primary benefit of inductive techniques is that the domain expert can provide what he is best at providing, i.e., tutorial examples and salient features of the domain, and not be called upon to generate an explicit declaration of his knowledge (Michie, 1982).

NATURAL-LANGUAGE INTERFACES

Many of the existing tools contain facilities for English-like interaction, smart editors, and the capability of handling multiple representations. Each one of these has advantages.

Natural-language interaction with users and builders provides the benefit that not all the details of the knowledge representation need to be known and manipulated for additions or modifications to be made to the system.

Smart editors can keep track of the system construction process and identify gaps, and they ease the entry of information. They can be fitted with a querying facility to relieve the builder of the system from some of the responsibility of having to know what the system needs to know.

They can also catch syntactic errors which occur during entry of information, thereby adding to the assurance that the system is receiving the knowledge it is supposed to receive.

Most of the tools and languages used for expert system construction include ways of mixing types of representation. This facilitates the construction of hybrid systems that can handle the complexities of real-world problems.

CLASSIFYING TOOL TYPES

Expert-system development tools and languages simplify and accelerate the representation and modification of knowledge, but there are no explicit knowledge-acquisition facilities in them. The principal difference between tools and languages is that the tools usually have a greater bandwidth for interacting with system developers and more preprogrammed options for representation and control. Some languages, such as Metalevel Representation System (MRS), facilitate the specification of metaknowledge and control, and have more flexibility, but also require more programming to construct the system.

The tools and languages used to build expert systems are media for expressing knowledge, but they are not tools for collecting. They assist in the building of expert systems primarily by setting the frameworks for the knowledge to enter. They can be restrictive if there is a mismatch between the built-in inference modes or the forms of representation and the problem solving structures and knowledge from the application domain.

CLASSIFYING TOOLS BY THEIR FUNCTION

KA tools can be classified several ways. The following paragraphs discuss many of the functions performed by KA tools. The possible applications of tools in these categories to the ASPA program are given in Box 1, and a list of tools that perform one or more of these functions is given in table 1.

BOX 1. APPLICATION OF TOOL CATEGORIES TO ASPA

DISCOVERY

Most current discovery tools require a base of fully analyzed cases. In ASPA, the only cases available are those in the Joint Munitions Effectiveness Manual (JMEM). None of the existing tools are directly applicable without some adjustment.

IDENTIFICATION

Identification tools would be useful for analytic subdomains in ASPA. The Expertise Transfer System (ETS) is not yet available and the Goal Directed Decision Structuring System (GODDESS) would need to be reimplemented. ROGET and MORE could be used to classify situations using intermediate features.

SELECTION OF REPRESENTATIONS

Selection tools do not exist yet, except possibly for some newly developed capabilities in Lenat's Representation Language Language (RLL). ROGET was to have been built to handle selection. However, it has only been expanded in the sense that a query is made to the knowledge engineer to ask what kind of system he feels is most like the one under construction. The only permissible answer to the query at this point is MYCIN.

TRANSFER

Transfer tools are numerous, but all are specialized for particular knowledge representations. Knowledge Base Interactive Classifier (KuBIC), which can construct Lisp Object Oriented Programming System (LOOPS) hierarchies, is unsupported and unavailable; its descendant, KLASSIC, may be available in source listing form.

VALIDATION

The System for Experimentation with Expert Knowledge (SEEK) requires a data base of cases; ONCOCIN and TEIRESIAS are parts of other applications; UNITS, CHECK, SEEK, and ATEST are incorporated into separate expert system building tools and environments which would require major effort to interface with ASPA.

Table 1. Principal uses of tools.

	Principal uses					
Tool	Disc	Ident	Select	Trans	Val	Note
ADVISE	X	X		X	X	a
ACLS	X	X		X		
AQ11	X			X		
ATEST					X	
BACON	X					
CHECK				X	X	
CLUSTER/2	X					
COBWEB	X					
DUCK				X	X	
ETS	X	X		X	X	a
EXPERT-EASE	X	X		X		
GEM	X					
GODDESS		X				a
ID3	X					
KAS				X	X	
KLASSIC				X		
KuBIC				X		a
METADENDRAL	X					
MORE		X		X	X	a
NANOKLAUS				X		a
ONCOCIN					X	a
RLL			X	X	X	
ROGET		X		X		a
RUMMAGE	X					
SALT		X		X		
SEEK	X				X	a
STAGGER	X					
TEIRESIAS	X			X	X	a
UNITS				X	X	

[a] See description in this report.

Legend.
Disc = discover knowledge
Ident = identify knowledge
Select = select knowledge representations
Trans = transfer knowledge
Val = validate knowledge
X = used

Discovery. Theoretically, many kinds of knowledge could be discovered by machine. In practice, very little of it is.

Present and future uses of discovery tools are listed below:

Present uses of discovery tools

- Reveal hidden relationships
- Find new concepts by regrouping information
- Learn from observations
- Predict the effects of actions
- Represent knowledge efficiently
- Reduce the difficulty of handling large volumes of information

Future uses of discovery tools

- Find new features
- Find new representations
- Find new concepts
- Find new techniques for solving problems

The METADENDRAL, CLUSTER/2, COBWEB, BACON, and GEM systems are examples of discovery tools.

Identification. The usual approach to knowledge acquisition begins with this function. The knowledge in question has been discovered, and is in use by the experts. Even if knowledge of a domain is readily available in this sense, determining what knowledge to use in the system can be very difficult. It might be expected that the experts will know what is important, but in practice they do not. They may know how to use their own knowledge, but they frequently do not know how to point to or characterize the knowledge they are using.

The ETS system is an example of a tool which contributes to the identification of knowledge.

Selection of representations. Tools for choosing representational formalisms do not yet exist. Knowledge engineers handcraft representations or select representation languages which they believe will be capable of organizing the knowledge for use in solving domain problems. Matching the kind of knowledge to the type of representation is an art. Some knowledge engineers, such as Greiner and Lenat, 1980, are attempting to codify their own knowledge and build tools capable of selecting a representation.

Transfer. Transfer tools make up the bulk of knowledge acquisition tools. They ease the process of moving preidentified knowledge into preselected representations by the following:

- Assisting in finding the appropriate place
- Catching misspellings

- Supplying defaults
- Doing type checking
- Catching syntax errors
- Maintaining consistent terminology
- Determining that incoming values are not out of range
- Assuring the consistent treatment of values
- Maintaining consistency between multiple copies of the same information which must reside in more than one place for the system to run with maximum efficiency
- Reminding system builders about the characteristics of the structure which is being filled out
- Signaling unusual forms
- Reporting back on the state of the transfer process

Validation. Validation tools find errors. Some tools check incrementally with each addition to the system, some work only when processing a complete prototype so that dynamic checks can occur while the system is running. The tools may be static or dynamic, incremental or all-at-once, feedback driven, or stand-alone. There are tools which check for consistency, completeness, redundancy, unused or unusable knowledge, unusual forms, and interactions among rules or heuristics. Some validation tools propose and perform experiments (e.g., SEEK2) to exercise the system. These tools attempt to determine that the knowledge in the system will function correctly and as expected.

CLASSIFYING TOOLS BY THE TYPES OF KNOWLEDGE USED TO ACQUIRE KNOWLEDGE

The knowledge used by a KA tool to acquire knowledge is ordinarily quite different from the knowledge it is designed to acquire. Thus, each knowledge acquisition tool can be classified by the kind of knowledge it uses as well as by the knowledge it acquires. The knowledge required by the tool may be quite extensive. Some of the most recently implemented tools, ROGET and MORE, draw upon knowledge of diagnostic problem solving, for example. The knowledge used in the tool guides subsequent acquisition of knowledge. Table 2 lists several categories of knowledge and the systems, shells, and languages using them.

Table 2. Types of knowledge used by existing shells and systems.

Type of Knowledge	Shell or System
Constructive Problem Solving and Debugging Techniques	SALT
Decision Analysis	GODDESS
Decision Tree Structuring	ACSL
Diagnostic Problem Solving	ROGET
Diagnostic Problem Solving and Casual Relations Model	MORE
Frame Representations and Documentation Procedures	UNITS
Hierarchical Structuring and Inheritance	KuBIC, NANOKLAUS
Hierarchical Structuring and Programming Techniques	RLL
Induction and Collections of Analyzed Cases	GEM, AQ11, ESEL
Induction, Chemistry, and Generate and Test Problem Solving	METADENDRAL
Induction and Medical Domain Metaknowledge from a Partially Constructed System	TEIRESIAS
Induction, Collections of Cases, and Metaknowledge about Production-Rule Structure Knowledge-Base Refinement	SEEK, ONCOCIN, ATEST
Personal Construct Psychology and Repertory Grid Technology	ETS
Semantic Network Structuring	KAS

The distinction between the knowledge used by the tool and the knowledge it acquires may be confusing because, at first glance, it seems logical that the first chunk of knowledge it acquired would form a base from which to obtain more. However, the current state of the art does not permit this in any general way. Only a few KA tools have even rudimentary capabilities of applying knowledge which they have acquired, and that knowledge is only a minute subset of what is acquired. In TEIRESIAS, for example, the structure of rules is acquired by a completely separate mechanism from the content of rules. The structure of rules is then applied by the system to improve the acquisition of the contents of rules. For a system built out of knowledge of diagnostic problem solving to improve itself analogously, it would have to be a learning program which discovers improved diagnostic problem solving models. A system built out of inductive methods would analogously need to learn new inductive techniques! It would be learning to learn. It is questionable to what extent even humans learn to learn.

The sense in which a KA tool uses the knowledge it acquires to obtain more is exemplified by a system which uses knowledge as it is acquired to fill in holes in a preselected structure; thereby detecting that only so much of the structure is still left to be filled in. The acquired knowledge is used just to assess the state of the knowledge acquisition process. More advanced tools may use the acquired knowledge to simulate the functioning of the partially constructed system to obtain feedback on its shortcomings for directing later efforts of acquiring knowledge. Again, the knowledge is functioning in a circumscribed way to build a performance subcomponent of the KA system.

Most older tools draw upon metaknowledge of the knowledge the tool was built to obtain. The PROSPECTOR Knowledge Acquisition System (KAS) is an example.

Once the metaknowledge is identified, a system to acquire domain knowledge can be built.

There are a variety of problem-solving paradigms, including the following:

- Diagnostic problem solving
- Analogical problem solving
- Constraint-satisfaction-problem solving
- Means-ends analysis
- Inductive-scientific-problem solving
- Debugging approaches (related to genetic algorithms and to the cycle of generating and testing candidate solutions or solution steps)

Some tools, even though domain-dependent, may still contain basic assumptions that are applicable to several domains. Once these assumptions are identified for one of the domains, they can be built into a knowledge acquisition system to collect information automatically in many of the others.

ADDITIONAL DIMENSIONS FOR CLASSIFYING TOOLS

In addition to classifications based on function and the kinds of knowledge needed, tools may be classified by a number of miscellaneous characteristics,

- The stage of knowledge acquisition to which the tool contributes.
- The kinds of knowledge it is capable of acquiring.
- The style of its user interface.
- The kind of user it is oriented toward, domain specialist or knowledge engineer.
- The type of knowledge already incorporated so that it functions to acquire the right knowledge.
- The knowledge acquisition goals that it satisfies.
- Who is replaced by it.
- The functions it performs.
- Practical considerations such as its availability, its developer; where, when, and if it is supported, what it's written in, what it runs on, and contacts.
- Whether it is static or dynamic, that is whether it operates while the system executes, or whether it only processes static structures and representations
- The kinds of knowledge representation schemes it uses.
- Whether or not the design goals are realistic.
- Whether or not the costs of building the tool outweigh the benefits of having and using it.
- Whether or not the knowledge that forms the basis of the tool is subject to change and is it grounded in methods (either domain specific or general) which will soon be outdated.
- Whether or not the tool allows for maintenance of the knowledge it will collect, or sets the collected information in concrete.
- Whether or not it can handle knowledge which is evolving and/or time-dependent.
- Whether or not it can collect adequate information to provide explanations for the knowledge it obtains.

- Whether or not it permits the identification of some knowledge as being uncertain.
- Whether or not it protects against clerical errors in entering information.
- Whether or not the human interface is natural and provides for a variety of representations.

ADVISE

At the University of Illinois, researchers are constructing an integrated knowledge acquisition and refinement system. The product of the effort, ADVISE, is a domain-independent general purpose inference system which can be taught in several different ways.

ADVISE does not restrict the user to a single knowledge representation and inference mechanism. This is in contrast to EMYCIN, for example, which allows only rule-based knowledge and a backward chaining control scheme. ADVISE employs three knowledge representations: a rule base, a conceptual network, and a relational data base.

Unlike typical rule bases, the rule base for ADVISE may be structured. Each node in the rule base structure may have contextual information associated with it, so rule groups provide another level of "knowledge chunking." The contextual information may include different rule evaluation settings which are accessed by the rule evaluator, and could also include TEIRESIAS-like rule models.

ADVISE includes many different modules for inference mechanisms. The inference mechanism language allows the user to define his control scheme in terms of already existing tools.

ADVISE has been used to develop three expert systems; PLANT/ds for diagnosing soybean diseases, PLANT/cd for predicting autumn damage to corn, and BABY for advising on neonatal intensive care.

ETS

The Expertise Transfer System (ETS) (Boose, 1984) uses interviewing methods from Kelly's (1955) Personal Construct Theory. Kelly was interested in helping people to categorize their experiences and classify their environment in a psychotherapeutic setting. The organizing principles result in structures that allow people to predict events more accurately and act more effectively, and a framework in which to reorganize knowledge to satisfy specific goals or needs (Shaw, 1981).

Kelly's theory is that each person tries to predict and control events by forming theories, testing hypotheses, and weighing experimental evidence. Among the techniques Kelly developed is the Repertory Grid Test for eliciting, listing, comparing, and rating role models.

Additional techniques for analyzing repertory grids have been developed, including an interviewing technique known as laddering, that connect a given concept to its subordinate concepts and superordinate concepts.

ETS automatically interviews the expert to obtain elements of the domain that need to be classified. By selecting triads of elements, a variety of traits are elicited by asking what distinguishes two of the members of the triad from the third. Laddering is used, as are several other methods, to help the expert expand on and verify the relationships between concepts. Subsequently, the system requests that the expert rate each element against each construct to create the rating grid.

The constructs are analyzed to determine which are nearly functionally equivalent, and to generate an entailment graph of implication relationships. The expert can then study the unexpected relationships revealed in the graph; and, with the help of the system, track down the original rating responses that were responsible for their appearance. Once the entailment arcs are corrected and accepted, the system again invokes the laddering method to try to refine any ambiguous construct relations into parallel or orthogonal forms.

These automated techniques avoid many of the inconsistencies and inaccuracies that arise from unstructured knowledge acquisition methods.

ETS generates production rules based on the expert's assessment of each construct's potential importance in solving problems in the domain.

The production rules are automatically assigned certainty factors. The expert must review these to be assured that they correspond with the relative strength of belief that he would associate with the rule. ETS generates the certainty factors by taking into account grid ratings, relative construct importance, and the certainty factor combination in either OPS-5 or KS-300, the two target languages for which ETS generates knowledge bases.

In addition to the basic rules derived from the rating grid, which are described as conclusion rules, ETS also generates intermediate rules based on the relations in the entailment graph. For each entailment, one rule is generated and a certainty factor based on the strength of the entailment is associated with it.

As a knowledge acquisition tool, ETS's strengths are in the feedback it provides to the expert and in its ability to detect and resolve ambiguities by interaction with the expert. Another benefit of ETS is the assistance it provides in the initial analysis of domain concepts; the expert is not entirely on his own.

While ETS could not be used to generate rules directly for ASPA, it could be used to elicit the initial traits and heuristics for some subareas of the ASPA domain. ETS is suited to analytic problems rather than synthesis-class problems. Its developers say that it does not readily handle combination problems which require both analysis and synthesis either, so it cannot be expected to obtain all the knowledge needed in ASPA. Nonetheless, ETS could be used for analytic subdomains which are found to be fairly independent during the decomposition of ASPA. Portions of ASPA involve synthesizing the results of several analytic components and ETS can be applied separately to the analytic components.

GODDESS

The Goal Directed Decision Structuring System, GODDESS, is an interactive tool that aids an expert in structuring and organizing his knowledge about a particular problem. By emulating a decision analyst, GODDESS elicits a goal-directed network of relationships from the decision maker.

GODDESS uses a structured English dialogue to construct a network that reserves structures for both actions and goals (including subgoals) or issues. Each action is characterized by two lists, a list of preconditions and a list of effects. The intent of each action is identified before the action is entered into the system, thus encouraging the user to discover novel alternatives. The expert is queried to obtain value judgements about the suggested subgoals so that future queries may be directed to the most important aspects of the domain. The value judgements can also contribute to evaluating the relative merits of different solution paths.

GODDESS is intended for use where manual interviewing would be either infeasible or uneconomical. It is geared toward the acquisition of the knowledge required for synthesizing plans. Since GODDESS is domain independent, its contributions to KA are primarily in the phases of identification and formalization. The domain knowledge is entered by the user for each separate specific problem solving situation so we would need to build a structure around this process to capture and integrate knowledge from several episodes of its use.

KuBIC

The Knowledge Base Interactive Classifier [KuBIC] (Finin and Silverman, 1984) is an interactive system that creates a hierarchical frame-structured knowledge base. It classifies concepts by asking questions that help to make the classification. If the expert provides enough information, KuBIC will simply classify the concept without any questions. So a user who is familiar with the knowledge base can use the system just as a classifier, while others may use the interaction to be presented with just those parts of the knowledge base which are relevant to the classification of the new concept.

The classifying process is divided into three phases:

1. Obtaining the initial description of the concept.

2. Finding the appropriate parent concept in the existing taxonomy.

3. Finding the appropriate immediate subconcepts in the existing taxonomy (which may be null).

Given a partially complete knowledge base, KuBIC asks questions to determine the most specific and most general subsumption relations between the new node and those already in the knowledge base.

The algorithm for KuBIC could be applied to knowledge representation systems or environments that contain classifiable knowledge bases such as LOOPS (Bobrow & Stefik, 1983), HPRL (Lanam et al., 1984), and KEE (Kehler & Clemenson, 1984). However, the

algorithm is currently applicable only to structures in which there is no procedural attachment. There is a descendent of KuBIC, with more "bells and whistles," called KLASSIC.

MORE

MORE (Kahn, Nowlan, & McDermott, 1984) and ROGET are tools based on a generic theory of diagnostic systems. MORE incorporates domain knowledge as it is obtained and uses this knowledge to detect inconsistencies or problems with the weights associated with rules. It starts with a model of domain relations, the structural relations within the domain. These include five types of entities:

- Hypotheses
- Symptoms
- Conditions
- Links
- Paths

NANOKLAUS

The Knowledge Learning and Using Systems (KLAUS) (Haas & Hendrix, 1983) use natural language dialogue to elicit a classification hierarchy from an expert. The expert must already have in mind the outline of the hierarchy for the acquisition to be successful.

The information that is acquired is that needed to support question- answering or fact-retrieval tasks that can be supplemented by the application of conventional software packages such as report generators, simulators, database management systems, and statistical packages.

The unique feature of the KLAUS systems are their capacity to simultaneously learn both the new domain concepts and the linguistic constructs used to express them. This form of knowledge acquisition also differs from acquiring knowledge for insertion into rule-based representations in that it is primarily the acquisition of facts.

The first pilot implementation of the KLAUS system, NANOKLAUS, prompts the user for the information it requires to assimilate new concepts.

ONCOCIN

The ONCOCIN system (Shortliffe et al., 1977) includes a tool which helps the expert identify problem areas in the knowledge base (Suwa, Scott & Shortliffe, 1982). Unlike the TEIRESIAS system, in which errors in the knowledge base are tracked down during problem-solving, the ONCOCIN system incorporates a tool which formally assesses the rules as they are initially entered into the knowledge base. This approach has the virtue of not being so dependent on chance to expose problems.

However, ONCOCIN's knowledge acquisition tool works only in the context of a single expert system, and requires that the knowledge base be already partially developed. It follows the usual technique for developing a knowledge base of generating one and testing its adequacy by use of fully analyzed examples. If any problems are exposed by the system's handling of the

examples, the knowledge base is modified. The system is then tested again and the cycle of generating and testing is repeated until the expert is satisfied with the system's performance.

In rule representations, conflict between rules can be recognized when two rules succeed in the same situation, but with conflicting results. Redundancy can be detected when two rules succeeding in the same situation have the same results or differ only in that one contains additional restrictions on the situations in which it succeeds. Incompleteness or gaps can be detected when a situation not covered by the antecedent of any rule requires some action or result. If it is possible to enumerate all of the situations for which a given result should ensue, then missing rules can be detected logically.

One way to generate a list of all the situations is to designate a special set of domain variables. Taking the values of these variables in every combination will define all possible situations.

ONCOCIN uses a technique for partitioning the knowledge base to obtain subsets of the original collection of rules. Since the partitions are based on which variable subsets appear in the rules, these can be checked independently for conflict, redundancy, subsumption, or missing rules.

ROGET

ROGET (Bennett, 1984), like MORE, is a tool to acquire the initial vocabulary and conceptual structure of a diagnostic problem solving domain. ROGET is guided by knowledge which knowledge engineers can provide about diagnostic problem solving in general, but it does not rely on domain-specific knowledge. Among the general categories of diagnostic tasks that it queries the user for are

- Determining problems
- Determining causes
- Recommending actions
- Determining additional tests
- Predicting observations
- Evaluating evidence

SEEK

A System for Empirical Experimentation with Expert Knowledge (SEEK) (Kulikowski, 1983) constructs, tests, and debugs a large collection of expert rules, and assists in rapidly updating and improving the performance of expert systems. It was developed in the context of a medical-diagnosis expert system, but applies more generally to classification problems, that is those problems which can be expressed as prespecified lists of conclusions and observations. SEEK is integrated into the general purpose system building tool EXPERT (Weiss & Kulikowski, 1979).

SEEK prompts the expert-system builder to enter descriptions of reasoning rules in the form of a table where conclusions, confidence levels, and necessary and exclusionary

requirements (along with some medical terminology) can be expressed; and automatically converts the tabular results into EXPERT rules.

If disagreements between the expert's conclusions and those of the expert system are detected by SEEK, it suggests experiments for resolving the disagreements. The suggestions take the form of alterations in the scopes of rules or groups of rules. SEEK employs various heuristics to prioritize the different specializations and generalizations of the rules based on expected improvements in performance. The system designer rates the experiments and makes the final choice of which experiments to run.

In SEEK, a database of problem-solving examples with independently validated conclusions are used to evaluate the results of experiments performed on the rules in the system. By perturbing the rules and assessing the number and type of resulting errors on the stored cases, it is able to suggest possible improvements. Using work by Politakis (Politakis and Weiss, 1982 and 1983) on statistical credit assignment it gathers statistics on the "missing components" that prevent rules from firing when they should.

During incremental development of an expert system, SEEK can be run repeatedly to verify that new knowledge is not adversely affecting the functioning of the original system and, if rule interactions do alter the model's conclusions so that they are at variance with the expert's, it makes suggestions for correcting the conflicts.

A new version of SEEK, SEEK2 (Ginsberg et al., 1985), extends SEEK's applicability to a more general class of knowledge bases. In addition, SEEK2 can function independently if knowledge base refinement is desired without human interaction.

TEIRESIAS

The TEIRESIAS system (Davis, 1976) provides an interactive, English-language front end to the MYCIN rule base. TEIRESIAS contains rule models which express metalevel knowledge about the forms of diagnostic and therapeutic rules. These models are used to generate expectations about the form and content of rules to be elicited from the experts which are then used to guide the debugging process.

TEIRESIAS is a domain-independent knowledge acquisition system for a fixed control structure and representation. Only the knowledge base can be changed, the system cannot alter the representation or the control. TEIRESIAS works to improve the content of the knowledge base, so the creation of the program has to be at a fairly advanced stage before it can be applied to the task of knowledge acquisition. Alternate applications of the tool are limited to systems which have the same identical control structure and the same representations.

Although the problem solving paradigm for ASPA does not match that for TEIRESIAS, it should be noted that analogous systems could be constructed for domains that contain ASPA-like representations and control structures. Constructing such a system would be a mammoth task and there is no guarantee that the techniques in TEIRESIAS will generalize and extend adequately to the different kinds of tasks found in ASPA.

AN IDEAL TOOL FOR KNOWLEDGE ACQUISITION

The ideal KA tool would include features to provide the following:

1. Aid in formulating, codifying, expressing, and modifying domain knowledge for insertion into knowledge-based systems.

2. Help in verifying that the knowledge is correct, consistent, and coherent, and help in detecting inconsistent usage of technical vocabulary.

3. Facilities to extract information to be used in explanations at a variety of levels.

4. Ability to obtain deep knowledge, compiled knowledge, heuristics and still-fluid experimental knowledge and label the knowledge collected with the appropriate circumstances, reliability, and instructions for its use.

5. Information extraction from multiple knowledge-sources such as textbooks and databases.

6. A natural interface for the expert or others who need to enter knowledge. During the earliest stages of knowledge acquisition it will know what questions to ask to maximize its performance.

7. Automation of the process of acquiring knowledge wherever possible.

8. Mechanisms for accepting advice and for converting that advice into useable representations in the knowledge based system to enhance the system's performance.

9. A method of obtaining knowledge about the limitations of the domain knowledge so that an expert system built by the KA tool could experience a graceful degradation of its functions at the periphery of its knowledge.

10. Automation of the choice of representations for the incoming information, or at least provide criteria for determining what representations are appropriate.

11. Links or other features to help bring the right knowledge to bear at the right time in the final system.

12. A querying method for missing information and gap detection in the knowledge already acquired. Such a tool would need to make use of knowledge in the domain as it is acquired, and the process of KA itself would improve as its knowledge base grew. However, the tool should not have to have substantial knowledge of the domain to begin the process of acquiring knowledge there.

13. Maintenance support of systems even if the domain were still evolving. Domain changes include changes in theories, changes in level of expertise, changes in the boundaries, changes in technology, the addition of new equipment, tests, devices, techniques, paradigms, scenarios, or new observations.

14. Removal procedures of acquired knowledge when, for example, that knowledge could not be validated. This would enable the knowledge to be extracted easily and cleanly without undue loose ends.

The ideal KA system should also be the following:

1. Understandable and able to make the knowledge in it understandable as well.

2. Acceptable to both the users and builders of expert systems, and its strengths and weaknesses should be known so that it can be used appropriately. Because knowledge which goes beyond the data is in principle not justifiable, the users and builders of knowledge bases will continue to be responsible for their validation.

3. Able to facilitate the inspection and evaluation of the knowledge it acquires. However, it could not be expected to take on the role of justifying the knowledge.

4. Able to make good use of domain experts, allowing them to do what they are best at, and refraining from requiring them to step out of their customary roles and into the roles of philosopher or analyst.

5. Nonconstrained in its ability to accept varied presentation of concepts and methods for modifying knowledge. Domain experts would be given the opportunity to present their knowledge in a variety of forms since constraints of expression might lead to the suppression of valuable components of the knowledge that do not fit the preconceived model of knowledge in the domain.

PROMISING IDEAS

TWO METHODS OF GATHERING AND STRUCTURING KNOWLEDGE

Plan schemas. A relatively straightforward way of gathering and structuring knowledge for ASPA would be to collect plan schemas along with rules for selecting them and filling in their details (instantiating them). The initial steps would be to construct plan schemas to build into the domain model. Subsequently, the procedures for selecting the right plan schema to elaborate for a specific problem would need to be elicited from domain specialists.

Schemas of derivations. A more powerful, but more complex, way of gathering and structuring knowledge for ASPA would be to obtain schematic derivations of plans. If the process of developing plans could be broken into a small, finite set of abstract derivational forms, knowledge about the domain could be used to select the appropriate schemas. Situation-specific information could be used to generate the appropriate derivation and produce the appropriate plan.

DERIVATION OF ENTAILMENTS AND PRODUCTION RULES

The philosopher David Hume (1739 and 1748) said causal connections are nothing but constant conjunctions of matters of fact. We should consider how deep we can go in obtaining knowledge by observing regularities in the matters of fact that are available to us.

Consider an example of a matrix of three cases (C1, C2, and C3) and two dichotomous variables (low-altitude delivery? and laydown delivery? where 1 = yes and 0 = no) expressing traits.

CASES	C1	C2	C3
Low?	1	0	0
Laydown?	1	0	1

In the first case, a delivery was both a low-altitude delivery and a laydown delivery, in the second case it was neither, and in the third it was a laydown delivery but not a low delivery. Since there are eight possible entailments between two trait pairs, we can systematically examine each possible relation and generate all possible rules which have not been eliminated by counterexamples.

A rule which would be generated by this tiny set of cases is "All low deliveries are (or should be?) laydown deliveries." (Because the traits are negations of each other, the other rule which is generated in this example is just the contrapositive of this one.)

In essence, this approach is an attempt to generalize from examples to generate rules of thumb, some of which may be ridiculous to the expert. The experts can react to these rules by coming up with additional missing examples to fill in a more comprehensive picture of the domain, or by qualifying the rule by indicating the more restrictive contexts in which the rule could function properly.

Another way to view the information gleaned by this process is as a collection of possible constraints which are to be validated by the experts before further processing can shape them into strategies for problem solving.

Variables which could appear in the cases include dichotomous pairs like "good for ___/not good for ___" or "useable for ___/not useable for ___," where the blanks are filled in with something like "night deliveries," or "vertical targets," or "surprise."

Each value of a nominal variable may be made into a separate two-valued variable to enter into the matrix. For example, the concept of weapon may be broken into the concepts of each particular weapon, so that there will be separate variables for "is or is not a MK82," "is or is not an APAM," etc., each of which is a dichotomy. Even variables with continuous values may be converted to some set of dichotomies for the purpose of this kind of analysis. For example, the variable "altitude" which ostensibly takes thousands of possible values could be broken into say three variables "low or not low altitude," "middle or not middle altitude," and "high or not high altitude."

CONSTRAINT SATISFACTION PROBLEMS

Planning problems may be interpreted as constraint satisfaction problems consisting of a finite set of variables, each of which has an associated finite set of possible values. A set of constraint relations identifies which of the values are mutually compatible for certain subsets of the variables. A solution to the planning problem is just an assignment of values to the variables which simultaneously satisfies all the problem constraints.

Note that not all variables are discrete, and other techniques may be appropriate for handling continuous variables which appear in constraint satisfaction problem-solving situations. It will suffice to create categories of values to restructure a continuous variable as a discrete one, but optimization methods are also available and are referred to in Appendix C, which briefly introduces the Karmarkar and Simplex algorithms.

During the planning stage, decisions are made about the values to assign to the variables. Each choice of a value may have effects on later decisions because of the narrowing of options as the planning process unfolds. The propagation of constraints must be passed to other parts of the planning process to reflect the limited options, and there must be a way to backtrack from ill-advised decisions during the early stages of planning when those selections of values preclude a solution.

One advantage of representing planning problems as constraint satisfaction problems is that some of the recent theoretical results on ordering the instantiations can be applied to recommend optimal approaches to the planning problems; ones which involve the least backtracking, ones which take advantage of the relative importance of different variables in meeting the constraints on the problem.

If the construction of solution plans can be made sufficiently simple, further efforts can be directed toward evaluating multiple plans, all of which satisfy the constraints on the problem, to select the best one on the basis of additional criteria.

Experts include as part of their expertise not just ways of handling complex choices, but also an ability to produce high-quality solutions over and above that required by the objective constraints on the solution. Heuristics obtained from experts should be clearly marked to indicate whether they pertain to achieving an acceptable solution, or whether they pertain to obtaining exceptionally good solutions.

USING DECOMPOSITION

In a domain which has been decomposed or structured by some process, the initial stages of knowledge acquisition can concentrate on subareas of the decomposition, and the most important information to acquire at an early stage will be that which identifies a situation as belonging to a particular place within the hierarchy or subarea. This will immediately constrain the search for solutions to a simpler, less voluminous collection of knowledge. The process of knowledge acquisition in such domains will be best oriented to devise methods to quickly differentiate subareas.

Different tasks can be isolated from separate stages in the planning process, and unique problem-solving methods may be applied depending on the type of task involved. Evidential reasoning is most appropriate for diagnostic tasks, causal deductive reasoning from axioms or first principles is most appropriate for consequence-finding tasks (sometimes called What-Will-Happen-If or WWHI tasks), and object-oriented programming hierarchical structures are most appropriate for data retrieval tasks which involve the inheritance or inference of values from data values in other concepts. The knowledge acquisition techniques most appropriate for these kinds of tasks also differ.

USING METARULES TO PARTITION A DOMAIN

Metarules can be generated automatically to partition a domain into manageable segments. The function of these metarules would be to focus attention to that segment of the knowledge base having rules with premises capable of evaluating the "true" given the current facts. If a particular conjunct appears in many different rule premises, evaluating this conjunct first and ordering the rules on the basis of this conjunct could improve the efficiency of traversing the rule space. Metarules generated to effect this ordering could take the form of ruling in a collection of rules or of ruling out some collection of rules, or both.

The value of adding a particular metarule to the rule collection can be calculated by examining the cost of evaluating that rule (a function of the number of conjuncts in its premise) with respect to the product of the chance of the premise being evaluated to true and the amount of pruning of the space of rules accomplished by the metarule.

VALIDATING KNOWLEDGE THROUGH FEEDBACK

Validating knowledge is a process of establishing that we agree with the knowledge; that it is placed where we want it to be, and is going to be used as we intended it to be. However, if we want to be assured that the expert's knowledge is what is represented in the system, there is no substitute for providing feedback directly to the expert. The feedback should not be a mere parroting of the expert's original input. It should reflect the intended use of the knowledge, the inferred interrelationships among elements of knowledge, and the control mechanisms for problem solving. Feedback can assist in correcting at least some cases where knowledge is erroneous, extraneous, inconsistent, or incomplete.

ALLOWING KNOWLEDGE ACQUISITION TO BE GUIDED BY ANALOGY

Where a given paradigm describes a range of problem-solving behaviors, a useful approach to acquiring detailed knowledge would be to probe for deviations from a simple representation of that paradigm. Automating knowledge acquisition for domains with this kind of knowledge might benefit from techniques being devised in current work on analogy in problem solving.

SEPARATING KNOWLEDGE AS IT IS ACQUIRED

Planning requires

1. Implementation methods for generating possible plans.

2. Control knowledge for guiding the search for satisfactory plans.

3. Causal knowledge for propagating constraints among subparts of the completed plan.

Knowledge acquisition techniques for ASPA should keep these separate so that possible plans can be identified on the basis of local considerations, leaving the identification of wise or optimal plans to global strategies at a different level.

USING HIGH-LEVEL KNOWLEDGE TO FACILITATE PLANNING

A taxonomy of partial plans can be used to limit search in the development of plans. Knowledge of features or behaviors common to all members of a class of partial plans provides constraints that can be introduced early in the hierarchical planning process. It is important to distinguish the specifications (what must be true) of the plan, from the known characteristics of the model consisting of the partially implemented plan (what is known to be true) since the latter provides constraints which must be propagated, while the former consists of the goals of the planning process.

CONCLUSIONS

BOTTLENECKS

Knowledge-acquisition has been described as the chief bottleneck in the development of expert systems. The aspects of knowledge acquisition that currently present the greatest difficulty are identification, representation, and validation.

IDENTIFICATION

The knowledge acquisition "bottleneck" resides in the knowledge identification phase of knowledge acquisition. Since identification of important knowledge requires a relevantly high level of understanding of the domain, knowledge engineers have had to acquire much of the expertise in a domain personally before they could incorporate it into an expert system.

REPRESENTATION

The matching of knowledge to its representation is not presently a good candidate for automation except insofar as the representation can be preselected for automatic entry of knowledge for a given subdomain.

VALIDATION

Knowledge validation is a pressing problem which is likely to get worse as inductive techniques proliferate to relieve the knowledge-identification bottleneck.

RECOMMENDATIONS

LONG TERM

For the long term, aspects of the problem of knowledge acquisition should be isolated and studied by means of small experiments from which general principles can be drawn. Combining the resulting general principles will enable us to ultimately construct robust systems that approach the ideal KA system described in an earlier section.

NEAR TERM

In the near term, ASPA can benefit most immediately by finding or developing techniques for knowledge identification, discovery, and validation.

The limited availability of most of the tools discussed in this report, and the mismatch between their original applications and ASPA, make it impractical to import them for use here. However, many of the insights that they contain can be used for knowledge acquisition in ASPA. Local reimplementation of some of the best ideas (notably those in ETS, GODDESS, and ONCOCIN) can allow us to tune the techniques to the exact needs of ASPA.

RECOGNIZING KNOWLEDGE TO SERVE AS A FOUNDATION FOR KA

Knowledge acquisition should itself be knowledge-based. The only knowledge that is available before any domain knowledge is collected is metaknowledge, e.g., knowledge of human psychology, of problem solving techniques, etc. As domain knowledge is acquired, additional metaknowledge will become available either through human inspection or by inductive analysis. Knowledge acquisition could then proceed at an accelerated pace.

USING ETS FOR IDENTIFICATION AND DISCOVERY

For assistance in knowledge identification and discovery, some of ETS's basic capabilities should be reimplemented. This may be unnecessary if ETS becomes available soon. The benefits of using ETS for obtaining knowledge in analytic subdomains of ASPA could be substantial.

CONSTRAINT RELATIONSHIPS

A technique for deriving constraint relationships between variable values should be implemented. Constraint graphs could be built using basic inductive techniques and data from domain specialists or sources such as JMEM.

BUILDING A TOOL FOR PLAN CONSTRUCTION

To address the synthetic side of ASPA plan construction, a tool should be developed to process information in constraint graphs. The tool would suggest optimal orderings for making decisions in constructing plans. Development of this tool would draw from recent work at UCLA by Rina Dechter and Judea Pearl on converting some constraint graphs to trees. The

suggested orderings for making decisions can be inspected by experts, thereby generating further knowledge acquisition possibilities.

DEVELOP A LOGICAL CHECKER FOR RULE VALIDATION

A static logical checker should be developed to assist in knowledge validation. This tool would warn of inconsistencies between rules and indicate when a rule subsumes or is subsumed by another. The tool would show in tabular form where objects, attributes, and values of attributes were referenced in rules, whether on the right or the left. The tables created by this tool would enable us to establish, using constraint relations, that all possible combinations of a number of domain variables are covered by the existing rules and, thereby, identify "theoretical" gaps in knowledge to check with the experts. The underlying assumption is that there should be a rule which applies in each situation defined by all possible combinations of the domain variables. Some combinations of the variables might not be meaningful, but a part of the validation process is to establish that fact.

CONSIDERATIONS FOR FUTURE RESEARCH AND DEVELOPMENT

There are a number of areas that should be considered for future expert systems even if there is little chance that the results could be applied to the ASPA project. The choices for constructing improved techniques for knowledge acquisition include two broad classes of approach. In one, the effort is to broaden the communication channel between the experts and the system, while on the other, both the knowledge engineer and the expert are eliminated in favor of permitting the system to discover knowledge on its own. Of course, combinations of the above alternatives would also be possible.

BROADENING CHANNELS OF COMMUNICATION

To broaden the channels of communications between domain experts and evolving expert systems, we could obtain or develop several items:

1. A smart editor
2. Improved metaknowledge of the domain
3. Static-knowledge analyzers
4. Execution-trace dialogues with experts

Smart editor. The construction of an editing facility which has built-in knowledge of the ASPA's internal representational formalisms to allow rapid, error-free entry of information into the knowledge-base cannot be expected to advance the field of known techniques for knowledge acquisition, and provide minor improvements over brute-force coding, especially in the context of the superior features of LOOPS.

Improved metaknowledge of the domain. The construction of a system with a comprehensive structural knowledge of the mission-planning domain presupposes that the metaknowledge reflecting this structure is known or is available to be collected first. We do not know enough about the domain of the air-strike planner to make use of metaknowledge of this kind in the early stages of KA for ASPA. Furthermore, we have no techniques for collecting it manually, and it may be even more difficult to acquire domain metaknowledge from the experts than to obtain domain knowledge. We do not expect to know enough about the structure on this level until the system is nearly complete. However, using the metaknowledge gathered for ASPA would be a viable approach for future systems to be built in related domains once ASPA has been completed, much as EMYCIN was useful as an expert system shell for similar domains after MYCIN was constructed.

Static-knowledge analyzers. The construction of a system to analyze the knowledge currently in ASPA would be quite valuable for validation of that knowledge. There is a variety of forms which the analyzer could take. At one level, all heuristic rules could be indexed by the concepts which appear in them, by specific values of the variables which they contain, or by their interactions where interactions could be defined in a lot of different ways. At another level, entailment relations among variables and known constraints among them could be spelled out for verification.

Execution-trace dialogues with experts. The construction of a system to support a dialogue with mission planners and system developers, using explanations of the behavior of the system drawn from traces of the inference paths, can be used to go beyond what static analyzers can accomplish for validation. This approach, which was taken in the TEIRESIAS system, is most suitable for the refinement of almost complete knowledge bases, and does not address the general problem of knowledge acquisition.

However, this kind of system is most valuable for knowledge validation, system maintenance, and enhancement, and needs to be considered seriously for ASPA if it is to remain useful over an extended period.

ALLOWING THE SYSTEM TO DISCOVER KNOWLEDGE ON ITS OWN

The second approach encompasses the possibilities of building a system to do the following:

1. Perform induction from fully analyzed cases
2. Perform induction by analogy
3. Contain problem solving metaknowledge
4. Perform induction from expert judgements

This approach includes possibly building a plan or partial plan generator and evaluator, and a hierarchical planning system with key cases for induction of heuristics.

Induction from analyzed cases. A system to perform induction from existing databases and test case files has been developed by Michalski at the University of Illinois. Michalski's system offers an arsenal of general purpose techniques that could be used to analyze typical cases and expert judgements. However, this system will only work in a domain in which numerous fully analyzed test cases are readily available or easily generated.

Plan or partial plan generator and evaluator. A system based on generate-and-test problem solving would be a useful tool, but it is not likely to be computationally tractable for ASPA unless it included heuristics to constrain searches. This means that much of the knowledge would already have to have been acquired before the system was built. Furthermore, the methods of representing plans and partial plans would have to be spelled out, and methods for evaluating plans would have to be developed.

Hierarchical planning system with key cases for induction of heuristics. Another alternative is the construction of a general purpose, layered, planning system within which repeated refinement of the specifications and key mission planning cases provide the necessary inputs for learning heuristics. Development of such a system would require some progress on theoretical research issues and it would necessitate the integration of research results from many places, particularly Pat Langley's work on SAGE (Langley, 1982) and work in an area called explanation-based generalization or induction by analytical generalization. This would be a large project.

Induction by analogy. The construction of a system which learns mission-planning techniques by analogy may be a poor approach judging from a comment of CDR Russell (1984) to the effect that it is better conceptually to start with a clean slate than with a standard plan which is then modified. Of course, analogy is a learning mechanism in addition to a problem-solving method, so analogy may still be an appropriate learning mechanism for the planning domain. However, if humans have problems learning in this domain by analogy, it is likely that there will be problems setting up a system to do so effectively. Another comment with respect to mission planning is that mission planners must learn by doing, not by watching. If this statement is correct, it may be difficult to set up the system to infer the planning processes by setting up traces of what the mission planner types in at a terminal, say.

Problem-solving metaknowledge. The construction of a system built on knowledge of problem-solving paradigms is another alternative for knowledge acquisition. In this case, the problem-solving framework determines what knowledge is needed from the domain.

The problem with this approach in general is that the problem-solving framework is not sufficiently specific to elicit detailed knowledge. However, systems built out of knowledge of problem solving types such as constraint satisfaction could be of use in subdomains of ASPA.

Induction from expert judgements. The construction of a system to perform induction on expert judgments was described above, but here I refer to not just judgements of cases, but microjudgements of the sorts available in documentation such as JMEM in tabular form. The information available in documents could provide a rich source of data for inductive techniques.

A PROPOSED KNOWLEDGE ACQUISITION TOOL

A tool could be constructed for ASPA that would receive as input the constraint relationships among domain concepts (variables) and produce as output "criticality" values to associate with concepts. These values may be mobilized to help organize the process of plan construction. They can be used to order planning decisions. They can be used in selecting variables and values within a variable for instantiation, a necessary process in filling out the details of a plan during planning. The criticality values may be incorporated into the object hierarchies and referred to as needed to determine rankings.

In addition, the criticality values could be built into future algorithms in several ways. They could be considered as default values, which could be overridden by local recalculations, or they could be used when final calculations of criticality are made for specific problems.

To generate the criticality values, designers of a general purpose KA tool can can draw upon the work of Sacerdoti (1974) on hierarchical planning and the work of Dechter (1985) and Pearl et al. (1983) on constraint graph properties and the possibilities of converting some constraint graphs to trees.

The users of this tool would be knowledge engineers and the tool's output can be used to prompt and challenge domain experts.

ACRONYMS

ACSL	Analogue Concept Learning System (forerunner of EXPERT-EASE)
ADVISE	General purpose inference system, a collection of tools for building expert systems
AGE	Attempt to Generalize
ART	Automated Reasoning Tool
ASPA	Air Strike Planning Advisor
ATEST	A knowledge-base refinement tool of the ADVISE system which generates testing examples, runs them under a variety of evaluation schemes, and checks for consistency and completeness
AQ11/AQ15	Inductive learning programs based on the Aq algorithm for producing minimal descriptions of classes
BACON	A family of concept-learning programs which can discover laws relating real-valued variables
CASNET	Causal Association Network
CHECK	Program to verify completeness and consistency (part of LES)
CLOT	Blood-Clot Consultation System
CLUSTER/2	A conjunctive conceptual clustering program
COBWEB	Conceptual clustering system that builds classification trees to maximize inference abilities using Gluck and Corter's category utility measure (Gluck and Corter, 1985)
DENDRAL	Organic Chemical Analysis System
EMYCIN	Essential MYCIN
ETS	Expertise Transfer System
EXPERT	Expert-system building tool with facilities to maintain test cases to repeatedly analyze the system's performance
EXPERT-EASE	Expert-system building tool for systems which perform classification or diagnosis

GEM	Generalization of Examples by Machine
ID3	Non-incremental concept learning system that builds sophisticated characterizations in the form of decision trees
IPS	Instructible Production System
JMEM	Joint Munitions Effectiveness Manual
K	Knowledge
KA	Knowledge Acquisition
KAS	Knowledge-Acquisition System for PROSPECTOR
KEE	Knowledge-Engineering Environment
KLASSIC	Elaboration of KuBIC
KLAUS	Knowledge Learning And Using Systems
KR	Knowledge Representation
KuBIC	Knowledge Base Interactive Classifier
LES	Lockheed Expert System
LISP	List Processing Language
LOOPS	Lisp Object Oriented Programming System
MDX	Medical Expert System that performs diagnoses related to cholestasis
METADENDRAL	Learning component for DENDRAL
MORE	Knowledge-acquisition system based on diagnostic problem solving
MRS	Metalevel Representation System (formerly Modifiable Representation System)
MYCIN	Infectious disease consultation system
NanoKLAUS	The first Knowledge Learning and Using System
ONCOCIN	Oncology Outpatient Monitoring System

OPS	A family of general-purpose production system languages, including OPS5 and OPS83
PRISM	Program for Research Into Self-Modifying systems
PROLOG	Programming in Logic
PROSPECTOR	Mineral Prospecting Expert System
PUFF	Pulmonary-Function Diagnosing System
RI	(now XCON)
RLL	Representation Language Language
ROGET	A knowledge-acquisition system for classification problem-solving tasks (named for the thesaurist)
ROSIE	Rule-Oriented System for Implementing Expertise
RUMMAGE	A hierarchical conceptual clustering program
SAGE	Strategy Acquisition by Generalization of Examples
SEEK	System for Experimentation with Expert Knowledge
STAGGER	Incremental concept learning system which can handle noise and track concept drift
TEIRESIAS	Interactive knowledge-base refinement tool for MYCIN
UCLA	University of California at Los Angeles
UNITS	Frame representation language tool
XCON	Expert Configurer of VAX 11/780 computer systems
XSEL	Expert Selling Assistant

REFERENCES

Aiello, N., and H.P. Nii (1981). "AGEPUFF: A Simple Event-Driven Program." Report HPP-81-25, Computer Science Dept., Stanford University, Stanford, CA.

Aiello, N., C. Bock, H.P. Nii,and W.C. White (1981). "Joy of AGE-ing: An Introduction to the AGE-1 System." Report HPP-81-23, Computer Science Dept., Stanford University, Stanford, CA.

Aikins, J.S., J.C. Kunz, E.H. Shortliffe, and R.J. Fallat (1983). "PUFF: An Expert System for Interpretation of Pulmonary Function Data." *Computers and Biomedical Research,* Vol. 16, pp. 199-208.

Alty, J.L., and M.J. Coombs (1984). "Associative and Casual Approaches to Diagnosis–INTERNIST and CASNET." *Expert Systems, Concepts and Examples,* Manchester, England: NCC Publications.

Bennett, J.S., and R.S. Engelmore (1984). "Experience Using EMYCIN." In B. Buchanan and E. Shortliffe (Eds.), *Rule-based Expert Systems,* Reading, MA: Addison-Wesley, pp. 314-328.

Bennett, J.S. (1984). "ROGET: Acquiring the Conceptual Structure of a Diagnostic Expert System." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems,* Denver, CO, 3-4 December.

Bennett, J.S. (1985). "ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System." *Journal of Automated Reasoning,* Vol. 1, No. 1, pp. 49-74

Bobrow, D.G., and M. Stefik (1981 and 1983). "The LOOPS Manual." Technical Report KB-VLSI-81-13, XEROX PARC.

Bobrow, D.G., and T. Winograd (1977). "An Overview of KRL, a Knowledge Representation Language." *Cognitive Science,* Vol. 1, No. 1, pp. 3-46.

Boose, J.H. (1984). "Personal Construct Theory and the Transfer of Human Expertise." *Proceedings of the National Conference on Artificial Intelligence,* Austin, TX, pp. 27-33.

Boose, J.H. (1985a). "A Knowledge Acquisition Program for Expert Systems Based on Personal Construct Psychology." *International Journal of Man-Machine Studies,* No. 23, pp. 495-525.

Boose, J.H. (1985b). "Rapid Acquisition and Combination of Knowledge from Multiple Experts in the Same Problem Domain." Boeing ATAD Technical Report BCS-G2010-23.

Boulanger, A.B. (1983). "The Expert System PLANT/CD: A Case Study in Applying the General Purpose Inference System ADVISE to Predicting Black Cutworm Damage to Corn." Report UIUCDCS-R-83-1134, Computer Science Dept., University of Illinois, Urbana, IL.

Bratko, I. (1986). *Prolog Programming for Artificial Intelligence.* Reading, MA: Addison-Wesley.

Brownston, L., R. Farrell, E. Kant, and N. Martin (1985). *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming.* Reading, MA: Addison-Wesley.

Buchanan, B.G., and E.A. Feigenbaum (1978) "Dendral and Metadendral; Their Applications Dimension." *Artificial Intelligence,* Vol. 11, No. 1-2, pp. 5-24.

Buchanan, B.G., D.R. Barstow, R. Bechtel, J.S. Bennett, W.J. Clancey, C.A. Kulikowsky, T.M. Mitchell, and D.A. Waterman (1983). "Constructing an Expert System." In F. Hayes-Roth, D.A. Waterman, and D.B. Lenat (Eds.), *Building Expert Systems.* Reading, MA: Addison-Wesley.

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

Chandrasekaran, B., and S. Mittal (1983). "Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems." In *Advances in Computers,* Vol. 22, New York, NY: Academic Press, pp. 218-295.

Chilausky, R., B. Jacobsen, and R.S. Michalski (1976). "An Application of Variable-Valued Logic to Inductive Learning of Plant Disease Diagnostic Rules." *Proceedings of the Sixth International Symposium on Multivalued Logic,* Utah.

Clayton, B.D. (1984). *ART Programming Primer.* Los Angeles, CA: Inference Corporation.

Clocksin, W.F., and C.S. Mellish (1982). *Programming in PROLOG.* New York, NY: Springer-Verlak.

Davis, R. (1980). "Metarules: Reasoning about Control." *Artificial Intelligence,* No. 15, pp. 179-222.

Davis, R. (1976). "Applications of Metalevel Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases." Report STAN-CS-76-552, Computer Science Dept., Stanford University, Stanford, CA.

Davis, R. (1978). "Knowledge Acquisition in Rule-Based Systems - Knowledge about Representation as a Basis for System Construction and Maintenance." In D.A. Waterman and R. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems,* New York, NY: Academic Press.

Davis, R. (1979). "Interactive Transfer of Expertise: Acquisition of New Inference Rules." *Artificial Intelligence,* Vol. 12, pp. 121-157.

Davis, R., and B.G. Buchanan (1977). "Metalevel Knowledge: Overview and Applications." *Proceedings of the 5th International Joint Conference on Artificial Intelligence,* Cambridge, MA, pp. 920-927.

Davis, R., B.G. Buchanan, and W. Shortliffe (1977). "Production Rules as a Representation for a Knowledge-Based Consultation Program." *Artificial Intelligence,* Vol. 8, No. 1, pp. 15-45.

Davis, R., and D. B. Lenat (1982). *Knowledge-Based Systems in Artificial Intelligence.* New York, NY: McGraw Hill.

Dechter, R. (1985). "Studies in the Use and Generation of Heuristics." Ph.D. Thesis, University of California, Los Angeles, CA, CSD-850033.

Duda, R., J. Gaschnig, and P. Hart (1979). "Model Design in the PROSPECTOR Consultant System for Mineral Exploration." In D. Michie (Ed.), *Expert Systems in the Micro-electronic Age,* Edinburgh, Scotland: Edinburgh University Press.

Duda, R.O., and R. Reboh (1984). "AI and Decision Making: the PROSPECTOR Experience." In W. Reitman (Ed.), *Artificial Intelligence Application for Business,* Norwood, NJ: Ablex.

Erman, L.D., P.E. London, and S.F. Fikas (1981). "The Design and Example Use of HEARSAY III." *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* Vancouver, BC.

Fain, J., D. Gorlin, F. Hayes-Roth, S.J. Rosenshein, H. Sowizral, and D. Waterman (1981). "The ROSIE Language Reference Manual." Technical Report N-1647-ARPA, Santa Monica, CA: The Rand Corporation.

Finin, T. Unpublished material on KLASSIC, University of Pennsylvania, PA.

Finin, T., and D. Silverman (1984). "Interactive Classification, a Technique for Building and Maintaining Knowledge Bases." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems,* Denver, CO, 3-4 December.

Fisher, D. (1984). "A Hierarchical Conceptual Clustering Algorithm." Technical Report, Computer Science Dept., University of California, Irvine, CA.

Fisher, D. (1986). "The Acquisition and Recognition of Basic Level Categories." Computer Science Dept., University of California, Irvine, CA.

Fisher, D., and P. Langley (1985). "Approaches to Conceptual Clustering." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp. 691-697.

Forgy, C.L., and J. McDermott (1977). "OPS, a Domain Independent Production System Language." *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* Cambridge, MA, pp. 933-939.

Forgy, C.L, (1981). "OPS5 Users Manual." Technical Report CMU-CS-81-135, Carnegie-Mellon University, Pittsburgh, PA, August.

Forgy, C.L. (1984). "The OPS83 Report." Carnegie-Mellon University, Pittsburgh, PA.

Freiherr, G. (1980). "The Seeds of Artificial Intelligence." Technical Report 80-2071, NIH-SUMEX-AIM, Bethesda, MD.

Gaines, B.R., and M.L.G. Shaw (1981). "New Directions in the Analysis and Interactive Elicitations of Personal Construct System." In Shaw, M.L.G. (Ed.), *Recent Advances in Personal Construct Technology,* New York, NY Academic Press.

Gaschnig, J. (1982). "PROSPECTOR: An Expert System for Mineral Exploration." In D. Michie (Ed.), *Introductory Readings in Expert Systems,* pp. 47-64, New York, NY: Gordon and Breach.

Genesereth, M.R. (1983a). "An Overview of Metalevel Architecture." *Proceedings of the National Conference on Artificial Intelligence,* Washington, DC, pp. 119-124.

Genesereth, M.R. (1983b). "The MRS Project." *The AI Magazine,* p. 89, Fall.

Genesereth, M.R. (1984). "An Overview of MRS for AI Experts." Memo HPP-82-27, Stanford University, Stanford, CA, January.

Ginsberg, A., S. Weiss, and P. Politakis (1985). "SEEK2: A Generalized Approach to Automatic Knowledge Base Refinement." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp. 367-374.

Gluck, M.A., and J.E. Corter (1985). "Information, Uncertainty, and the Utility of Categories." *Proceedings of the Seventh Annual Conference of the Cognitive Science Society,* Irvine, CA, pp. 283-287.

Goldstein, I.P., and R.B. Roberts (1979). "Using Frames in Scheduling." In P. Winston and R. Brown (Eds.), *Artificial Intelligence: An MIT Perspective,* Vol. 1, Cambridge, MA: MIT Press, pp. 253-284.

Greiner, R., and D.B. Lenat (1980a). "Details of RLL-1." Report HPP-80-23, Stanford University, Stanford, CA, October.

Grenier, R.D., and D.B. Lenat (1980b). "RLL: A Representation Language Language." *Proceedings of the First National Conference on Artificial Intelligence,* Stanford CA, pp. 165-169.

Grover, M.D. (1983). "A Pragmatic Knowledge Acquisition Methodology." *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany. pp. 436-438.

Haas, N., and G. Hendrix (1983). "Learning by Being Told: Acquiring Knowledge for Information Management." In R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: an Artificial Intelligence Approach,* Palo Alto, CA: Tioga Press.

Hart, P.E., and R.O. Duda (1977). "PROSPECTOR–A Computer Based Consultation System for Mineral Exploration." Technical Note 155, SRI-AI, October.

Hayes-Roth, F., and D.L. Mostow (1975). "Collected Papers on the Learning and Recognition of Structured Patterns." Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA.

Hayes-Roth, F., D.A. Waterman, and D.B. Lenat (Eds.) (1983). *Building Expert Systems.* Reading, MA: Addison-Wesley.

Hayes-Roth, F., P. Klahr, and D.J. Mostow (1981). "Advice-Taking and Knowledge Refinement: an Iterative View of Skill Acquisition." In J.A. Anderson (Ed.), *Skill Acquisition and Development,* Hillsdale, NJ: Lawrence Erlbaum.

Holtzman, S. (1981). "A Model of the Decision Analysis Process." Dept. of Engineering-Economic Systems, Stanford University, Stanford, CA.

Holtzman, S. (1984). "Academic Economics." Dept. of Engineering-Economic Systems, Stanford University, Stanford, CA.

Howard, R.A. (1968). "The Foundations of Decision Analysis." In *Readings in Decision Analysis,* Second Edition, SRI International, Menlo Park, CA, 1977, reprinted from *IEEE Transactions on Systems Science and Cybernetics,* Vol. SSC-4, No. 3, pp. 1-9, September.

Howard, R.A. (1973). "Decision Analysis in Systems Engineering." In *Readings in Decision Analysis,* 2nd Ed., SRI International, Menlo Park, CA, 1977, reprinted from Ralph F. Miles (Ed.), *Systems Concepts: Lectures on Contemporary Approaches to Systems,* New York, NY: John Wiley and Sons.

Hume, D. (1739). *Treatise of Human Nature,* Book 1, part III, section XIV, and (1748) *Enquiry Concerning Human Understanding,* part II, section VII.

Kahn, G., S. Nowlan, and J. McDermott (1985). "MORE: An Intelligent Knowledge Acquisition Tool." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp. 581-584.

Kahn, G., S. Nowlan, and J. McDermott (1984). "A Foundation for Knowledge Acquisition." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems,* Denver, CO, 3-4 December.

Keeney, R.L. and H. Raifa (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York, NY: John Wiley and Sons.

Kehler, T.P., and G.D. Clemenson (1984). "An Application Development System for Expert Systems." *Systems and Software,* January.

Kelly, G. (1955). *The Psychology of Personal Constructs.* New York, NY: Norton.

Kelly, G.(1963). "Non-parametric Factor Analysis of Personality Theories." *Journal of Individual Psychology,* 19.

Kulikowski, C.A. (1983). "Knowledge Acquisition and Learning in EXPERT." *Proceedings of the 2nd International Machine Learning Workshop,* Allerton House, University of Illinois, Urbana, IL, June 22-24. pp. 71-73,

Kunz, J.C., T.P. Kehler, and M.D. Williams (1984). "Applications Development using a Hybrid AI Development System." *The AI Magazine,* Vol. 5, No. 3, Fall.

Lanam, D., R. Letsinger, S. Rosenberg, P. Huyun, and M. Lemon (1984), "Guide to the Heuristic Programming and Representation Language Part 1: Frames." Technical Report AT-MEMO-83-3, Application and Technology Laboratory, Hewlett-Packard Company, Palo Alto, CA, June.

Langley, P. (1982). "Strategy Acquisition Governed by Experimentation." CIP Working Paper 431, Carnegie-Mellon, Pittsburgh, PA.

Langley, P. (1983a). "Learning Effective Search Heuristics." *Proceedings of the Eighth International Joint Conference of Artificial Intelligence,* Karlsruhe, West Germany, pp. 419-421.

Langley, P. (1983b). "Exploring the Space of Cognitive Architectures." *Behavioral Research Methods and Instrumentation,* 15, pp. 289-299.

Langley, P. (1985). "Learning to Search From Weak Methods to Domain-Specific Heuristics." *Cognitive Science,* 9, pp. 217-260.

Langley, P., G.L. Bradshaw, and H.A. Simon (1983). "Rediscovering Chemistry with the BACON System." In R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Palo Alto, CA: Tioga Press, 307-329.

Langley, P., J. Zytkow, H.A. Simon, and G.L. Bradshaw (1986). "The Search for Regularity: Four Aspects of Scientific Discovery." In R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: an Artificial Intelligence Approach,* Vol. 2, Los Altos, CA: Morgan-Kaufman Publishers, pp. 425-469.

Langlotz, C., P., and E.H. Shortliffe (1983). "Adapting a Consultation System to Critique Users' Plans." *International Journal of Man-Machine Studies,* Vol. 19, pp.479-496.

Lindsay, R.K., B.G. Buchanan, E.A. Feigenbaum, and J. Lederberg (1980). "Applications of Artificial Intelligence for Organic Chemistry." In *The Dendral Project,* New York, NY: McGraw-Hill.

March, J. (1978). "Bounded Rationality, Ambiguity, and the Engineering of Choice," *Bell Journal of Economics,* Vol. 9, p. 587, October.

Marcus, S., J. McDermott, and T. Wang (1985). "Knowledge Acquisition for Constructive Systems." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp. 637-639.

Matheson, J.E., and R.A. Howard (1977). "An Introduction to Decision Analysis." In *Readings in Decision Analysis,* 2nd Ed., Menlo Park, CA: SRI International.

McCarthy, J. (1980). "Circumscription–a Form of Non-Monotonic Reasoning," *Artificial Intelligence,* 13, pp. 27-39.

McDermott, D.V. (1984). "Duck Builds Intelligent Systems." *Applied Artificial Intelligence Reporter,* Vol. 2, No. 2, November.

McDermott, D.V. (1985). "The Duck Manual." Technical Report 399, Computer Science Dept., Yale University, New Haven, CT, June.

McDermott, J. (1981). "R1: The Formative Years." *AI Magazine,* Vol. 2, No. 2, pp. 21-29.

McDermott, J. (1982a). "R1: A Rule-Based Configurer of Computer Systems." *Artificial Intelligence,* Vol. 19, No. 1, pp. 39-88.

McDermott, J. (1982b). "XSEL: A Computer Salesperson's Assistant." In *Machine Intelligence 10,* Chichester, UK: Ellis Horwood Ltd., pp. 325-337.

McDermott, J. (1984). "Building Expert Systems." In W. Reitmen (Ed.), *Artificial Intelligence Applications for Business,* Norwood, NJ: Ablex.

Michalski, R.S. (1980). "Pattern Recognition as Rule-Guided Inductive Inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-2, No. 2, 3, 4, pp. 349-361.

Michalski, R.S., and A.B. Baskin (1983). "Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: the ADVISE System." *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pp. 256-258.

Michalski, R.S., A.B. Baskin, M.R. Seyler, and A.B. Boulanger (1984). "A Technical Description of the ADVISE Meta-expert System." Internal report, Intelligent Systems Group, Computer Science Dept., University of Illinois, Urbana, IL.

Michalski, R.S., and R.L. Chilausky (1980a). "Knowledge Acquisition by Encoding Expert Rules Versus Computer Induction from Examples: A Case Study Involving Soybean Pathology." *International Journal for Man-Machine Studies,* No. 12, pp. 63-87.

Michalski, R.S., and R.L. Chilausky (1980b). "Learning by Being Told and Learning from Examples: an Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis." *International Journal of Policy Analysis and Information Systems,* Vol. 4, No. 2.

Michalski, R.S., J.H. Davis, V.S. Bisht, and J.B. Sinclair (1983). " A Computer-Based Advisory System for Diagnosing Soybean Diseases in Illinois." *Plant Disease,* April.

Michalski, R.S., and R. E. Stepp (1983). "Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI, Vol. 5, No. 4, pp. 396-410.

Michie, D. (1982). "Experiments on the Mechanism of Game Learning." *Computer Journal,* Vol. 25, No. 1, pp. 105-112.

Miller, P.L. (1984). *A Critiquing Approach to Expert Computer Advice: ATTENDING,* Marshfield, MA: Pitman.

Mitchell, T.M. (1978). Version Spaces: an Approach to Concept Learning," Ph.D. Dissertation, Stanford University, Stanford, CA, December.

Mitchell T.M., P.E. Utgoff, and R. Banerji (1983). "Learning by Experimentation: Acquiring and Modifying Problem Solving Heuristics." Computer Science Research Laboratory, LCSR-TR-31, Rutgers University, 1982. (Also a chapter in R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Palo Alto, CA: Tioga Press, pp. 163-190.

Mittal, S., and C.L. Dym. (1985). "Knowledge Acquisition from Multiple Experts." *The AI Magazine,* Vol. 6, No. 2, pp. 32-36, Summer.

Nii, H.P., and N. Aiello (1979). "AGE (Attempt to Generalize): a Knowledge-Based Program for Building Knowledge-Based Programs." *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* Tokyo, Japan, pp. 645-655.

Nordhausen, B. (1986). "Conceptual Clustering using Relational Information." Computer Science Dept., University of California, Irvine, CA, 1986. *Proceedings of the Fifth National Conference on Artificial Intelligence,* pp. 508-512.

Nguyen, T.A., W.A. Perkins, T.J. Laffey, and D. Pecora (1985). "Checking an Expert Systems Knowledge Base for Consistency and Completeness." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp. 375-378.

Ohlsson, S., and P. Langley (1986). "PRISM Tutorial and Manual" (draft). University of California, Irvine, CA

Paterson, A. (1983). "An Application of CLUSTER Learning Programs to the Synthesis of Decision Structures for the KPK Chess Endgame." Technical Report VIVCDCS-R-83-1156, Computer Science Dept., University of Illinois, Urbana, Il.

Paterson, A., and T. Niblet (1982). "ACLS User Manual." Report, Intelligent Terminals, Ltd., 15 Canal Street, Oxford, UK, OS26BH.

Pearl, J., A. Leal, and J. Saleh (1982). GODDESS: a Goal-directed Decision Structuring System." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-4. No. 3, pp. 250-262 May.

Perrone, J. (1983). "Expert Systems Get Personal: Modeling Expertise with the IBM PC." Report, Jeffery Perrone and Associates, Inc., 3685 17th Street, San Francisco, CA.

Politakis, P.G. (1982). "Using Empirical Analysis to Refine Expert System Knowledge Bases." Ph.D. Thesis, Rutgers University Computer Science Research Laboratory, CBM-TR-130, New Brunswick, NJ.

Politakis, P.G., and S.M. Weiss (1982). "A System for Empirical Experimentation with Expert Knowledge." *Proceedings of the Fifteenth Hawaii International Conference on Systems Sciences,* Honolulu, HI, pp. 649-657.

Politakis, P.G., and S.M. Weiss (1984). "Using Empirical Analysis to Refine Expert System Knowledge Bases." *Artificial Intelligence,* Vol. 22, No. 1, pp. 23-48.

Quinlan, J.R. (1979). "Discovering Rules by Induction from Large Collections of Examples." In D. Michie (Ed.), *Expert Systems in the Micro-Electronic Age,* Edinburgh, Scotland: Edinburgh University Press.

Quinlan, J.R. (1983a). "Learning Efficient Classification Procedures and Their Application to Chess End Games." In R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: an Artificial Intelligence Approach,* Palo Alto, CA: Tioga Press. pp. 463-482.

Quinlan, J.R. (1983b). "Learning from Noisy Data." *Proceedings of the Second International Machine Learning Workshop,* pp. 58-64, Urbana, IL.

Quinlan, J.R. (1986). "Induction of Decision Trees." *Machine Learning,* Vol. 1, No. 1, pp. 88-106.

Reboh, R. (1981). "Knowledge Engineering Techniques and Tools in the PROSPECTOR Environment." Technical Note 243, Stanford Research Institute, Menlo Park, CA.

Reboh, R. (1979). "The Knowledge Acquisition System." In R.O. Duda (Ed.), *A Computer-Based Consultant for Mineral Exploration,* Stanford Research Institute, Artificial Intelligence Center, Menlo Park, CA.

Reinke, R.E. (1983). "PLANT/ds: An Expert System for Diagnosing Soybean Diseases Common in Illinois. User's Guide and Program Description." Report No. UIU-CDCS-F-83-912, Computer Science Dept., University of Illinois, Urbana, IL.

Robinson, J.A. (1979). *Logic: Form and Function, The Mechanization of Deductive Reasoning.* New York, NY: North Holland.

Rodewald, L.E. (1984). "BABY: An Expert System for Patient Monitoring in a Newborn Intensive Care Unit." M.S. Thesis, Computer Science Dept., University of Illinois, Urbana, IL.

Rosenberg, S. (1983). "HPRL: a Language for Building Expert Systems." In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany, pp. 215-217.

Rosenbloom, P.S., J.E. Laird, J. McDermott, A. Newell, and E. Orciuch (1984). "R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem-solving Architecture." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems,* Denver, CO, pp. 65-72.

Russell, D. (1984). Transcription of an ASPA Project tape, Naval Ocean Systems Center, San Diego, CA.

Rychener, M.D. (1981). "Approaches to Knowledge Acquisition: the Instructable Production System Project." Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA.

Rychener, M.D. (1983). "The Instructible Production System: A Retrospective Analysis." In R.S. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: an Artificial Intelligence Approach,* Palo Alto, CA: Tioga Press, pp. 429-459.

Sacerdoti, E.D. (1974). "Planning in a Hierarchy of Abstraction Spaces." *Artificial Intelligence,* Vol. 5, No. 2, pp. 115-135.

Sacerdoti, E.D. (1975). "A Structure for Plans and Behavior." Technical Note 109, SRI-AI, August.

Schlimmer, J.C., and D. Fisher (1986). "A Case Study of Incremental Concept Induction." Computer Science Dept., University of California, Irvine, CA, 1986. *Proceedings of the Fifth National Conference on Artificial Intelligence,* pp. 496-501.

Schlimmer, J.C., and R. H. Granger, Jr. (1986). "Incremental Learning from Noisy Data." *Machine Learning,* Vol. 1, pp. 317-354.

Shortliffe, E.H. (1976). *Computer-Based Medical Consultations: MYCIN,* New York: Elsevier/North-Holland.

Shortliffe, E.H. (1981). "ONCOCIN, an Expert System for Oncology Protocol Management." *Proceedings of the Seventh International Joint Conference* on Artificial Intelligence, Vancouver, BC, pp. 876-881.

Silverman, D.L.(1984). "An Interactive, Incremental Classifier." Technical Report MS-CIS-84-10, University of Pennsylvania, Philadelphia, PA.

Simon, H.A. (1977). *The New Science of Management Decision,* Revised Ed. Englewood Cliffs, NJ: Prentice Hall, Inc.

Simon, H.A. (1983). "Search and Reasoning in Problem Solving." *Artificial Intelligence,* Vol. 21, No. 1-2, pp. 7-29.

Shaw, M.L.G. (1980). *On Becoming a Personal Scientist.* New York, NY: Academic Press.

Shaw, M.L.G. (Ed.) (1981). *Recent Advances in Personal Construct Technology.* New York, NY: Academic Press.

Shaw, M.L.G. (1984). "Interactive Knowledge Elicitation." *Proceedings of the Canadian Information Processing Society Annual Conference,* Calgary, AB.

Slater, P. (Ed.) (1977). *Dimensions of Interpersonal Space.* Vol. 2, London, England: Wiley.

Smith, R.G., and J.D. Baker (1983). "The Dipmeter Advisor Systems, a Case Study in Commercial Expert System Development." *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany, pp. 122-129.

Stefik, M. (1979). "An Examination of a Frame-structured Representation System." *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* Tokyo, Japan, pp. 845-852.

Stefik, M., D.G. Bobrow, S. Mittal, and L. Conway (1983). "Knowledge Programming in LOOPS: Report on an Experimental Course." *The AI Magazine,* pp. 3-13, Fall.

Strang, G. (1986). *Introduction to Applied Mathematics.* Cambridge, MA: Wellesley-Cambridge Press, pp. 673-689.

Suwa, M., A.C. Scott, and E.H. Shortliffe. (1982). "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System." *The AI Magazine,* Vol. 3, No. 4, pp. 16-21, Fall.

Tversky, A., and D. Kahneman (1981). "The Framing of Decisions and the Psychology of Choice." *Science,* Vol. 211, pp. 453-458, January.

van Melle, W., E.H. Shortliffe, and B.G. Buchanan (1981). "EMYCIN: a Domain-Independent System that Aids in Constructing Knowledge-Based Consultation Programs." *Machine Intelligence,* Infotech State of the Art Report, Series 9, No. 3, pp. 249-263.

van Melle, W., E.H. Shortliffe, and B.G. Buchanan (1984). "EMYCIN: a Knowledge Engineer's Tool for Constructing Rule-based Expert Systems." In B.G. Buchanan and E.H. Shortliffe (Eds.), *Rule-based Expert Systems,* New York, NY: Addison-Wesley, pp. 302-328.

Walker, A. (Ed.), M. McCord, J.F. Sowa, and G. Wilson (1987). *Knowledge Systems and PROLOG: a Logical Approach to Expert Systems and Natural Language Processing,* Reading, MA: Addison-Wesley.

Weiss, S., C. Kulikowski, S. Amarel, and A. Safir (1978). "A Model-Based Method for Computer-Aided Medical Decision-Making." *Artificial Intelligence,* Vol. 11, No. 1-2, pp. 145-172,

Weiss, S.M., and C.A. Kulikowski (1979). "EXPERT: A System for Developing Consultation Models." *Proceedings of the Sixth International Joint Conference on Artificial Intelligence,* Tokyo, Japan, pp. 942-947.

Weiss, S. M., K.B. Kern, and C.A. Kulikowski (1980). "A Guide to the Use of the EXPERT Consultation System." Technical Report CBM-TR-94, Computer Science Dept., Rutgers University, New Brunswick, NJ.

Weiss, S.M., and C.A. Kulikowski (1981). "Expert Consultation Systems: the EXPERT and CASENET Projects." In A.H. Bond (Ed.), *Machine Intelligence,* Infotech State of the Art Report, Series 9, No. 3, Permagon Infotech Limited, pp. 339-353.

Weiss, S.M., and C.A. Kulikowski (1984). *A Practical Guide to Designing Expert Systems.* NJ: Rowman and Allenheld.

Weiss, S., C. Kulikowski, C. Apte, and M. Uschold (1982). *Building Expert Systems for Controlling Complex Programs.* Rutgers University Computer Science Research Laboratory, LCSR-TR-40, New Brunswick, NJ.

Weiss, S., C. Kulikowski, and B. Nudell, (1979). *Learning Production Rules for Consultation Systems.* Rutgers University Computer Science Research Laboratory, No. 158, New Brunswick, NJ.

Williams, C. (1984). ART: The Advanced Reasoning Tool. Inference Corporation Report, Inference Corp., Los Angeles, CA.

APPENDIX A

METAKNOWLEDGE

A complete description of the kinds of knowledge that can be expected to need representation in an expert system would be very long indeed. Following is a chart of types of knowledge, along with an annotation to indicate who needs to know the knowledge of that particular type, just to indicate the extent of the possible types of knowledge. The annotation in the first column is to be interpreted as follows:

A - The knowledge acquirer must know.

E - The knowledge engineer must know.

S - The system must know.

Lower case indicates that it is not necessary for them to know, although it would be nice if they did. An omitted annotation indicates that they need not know.

Who Need Know		Kind of Metaknowledge
aes	1.	K of how to solve problems (K of how to make other K useful). (This is K common to all domains and to all problems.)
aes	2a.	K about the importance of other K (possibly with respect to a variety of factors, including goals, etc.).
aes	2b.	K about the relative importance of other K (again possibly with respect to a variety of factors, including goals, etc.).
aes	2c.	K about the reasons for the importance of other K
AES	3.	K about the reliability of other K.
Aes	4.	K about the sources of other K and pertinent details about those sources. Types of sources include: a. Written vs oral. b. Texts, notes, memos, official guidelines or communications, etc. c. Tables, charts, drawings, maps, databases, etc. d. Common sense.

e. Experience.

f. Hearsay.

g. Courses or tutorials.

5. K about the limitations of current K (K about the extent of and type of ignorance).

aEs a. K about what K is circumscriptively give-up-able (Columbus's egg solution). (This includes assessing what in the statement of the problem may be inaccurate and may need modification before an attempt is made to solve the problem.)

ae b. K about the tenuousness of present theoretical frameworks (Neurath's boat).

Es c. K about what is combinatorially too inefficient to find out even though it follows in principle from current K.

aEs d. K about the aptness of the model of the domain and, hence, about the range of applicability of current K to the real world.

aES e. K that our model of the domain is inadequate in some respect(s) (e.g., Ptolemaic model of the universe is inadequate in explanatory power even though solutions to problems could be devised in that model).

aEs f. K that we do not have a model of some aspect of the real world.

e g. K about the limitations of formalisms and how that effects what it is possible to automate.

ae 6. K about the domain context of current K, e.g.:

a. Structure (hierarchical, etc.).

b. Complexity.

c. Type of boundary for the domain (fuzzy, amorphous, invisible, or crisp, clean, exact boundaries).

d. Homogeneity of the domain (on a variety of parameters).

e. Axiomatizability of theories or models of the domain.

f. Levels of abstraction in the domain.

Es 7. K about the current problem's context and the aptness of the current formulation of the problem.

Es 8. K of the range of possible reformulations and restatements of the current problem (the current description of the problem may make it considerably more difficult [or even impossible] to solve).

eS 9. K of the state of the problem solving effort, e.g.:

a. Its progress.

b. Its goals, their precedence relations, and which ones remain to be satisfied.

c. Its milestones.

es 10. K of the branching factor for this problem (important for efficient search - a search space with a high branching factor is more suited to backwards chaining and other branching structures can be best suited to forward chaining or even mixed chaining approaches).

AES 11. K of the place in the model or other theoretical framework of a given bit of current K.

AEs 12. K of the interrelationships of other K.

AES 13. K of constraints on the solutions acceptable (e.g. official standards prohibiting certain otherwise reasonable solutions).

aES 14. K about what K does not change after an action has occurred.

aEs 15. K about the representation of objects in the system.

aEs 16. K about the representation of functions in the system.

aEs 17. K about the representation of actions.

aEs 18. K about the representation of the effects of actions on the world.

aEs 19. K about the representation of time, duration, and simultaneity.

aEs 20. K about the representation of inference rules.

aEs 21. K about the representation of reasoning strategies.

es 22. K about how to abstract from current K to more general K.

es 23. K about limitations of the abstractable K.

aes 24. K for directing the application of given K.

Aes 25. K about how to accumulate more K.

aEs 26. K about what aspects of current K need constant updating.

aEs 27. K about how to maintain K.

aEs 28. K about the human interface for the system and how to present K to the human user.

aes 29a. K about the particular [current] user.

aes 29b. K about the user's:

a. Needs.

b. Style of interaction.

c. Level of expertise.

d. State of comprehension of the context of the current problem.

aEs 30. K about what problems are beyond the system's scope. (A system which possesses this kind of K can gracefully back out of tackling problems rather than falsely producing "solutions" which are not identified in any way as being of poor quality.)

aes 31. K of what K is explanatory material for answering certain questions (K of how to construct explanations, how to use K in explanations).

es 32. K of how to instantiate general K (K of the range of values permissible for the variables in generalities. This is the mathematical notion of domain).

aEs 33. K of time constraints on producing a solution to the problem.

aes 34. K of what K to focus on to achieve a solution during different stages of the problem solving effort.

aes 35. K of what K can be obtained by observation and what K must be inferred or indirectly detected.

Es 36. K of all data structure relationships on which the system architect's design depends.

aes 37. K of the types of measurement in the domain, e.g., are the scales of measurement:

a. Nominal?

b. Ordinal?

c. Interval?

d. Ratio?

aes 38. K of the dimensionality of scales used in the domain for measurement.

aes 39. K of expected measurement error.

Es 40. K of resources available to be expended on generating a solution to the problem.

Es 41. K about the representation of states.

Es 42. K about the representation of procedures.

Es 43. K about the representation of resources.

Es 44. K about the representation of activities.

Es 45. K about the representation of schedules, milestones, and time windows.

Es 46. K about the representation of goals.

Es 47. K about the representation of plans.

Es 48. K about the representation of causality.

aEs 49. K about the scope of applicability of other K.

a. Temporal.

b. Logical.

c. Problem situation specific.

e 50. K about frequency of use of other K.

ae 51. K about who discovered or detected the truth of a current piece of K (authority for a current bit of K).

Es 52. K about the representation of priorities.

Es 53. K about the value of applying a current piece of K to problem solving.

aes 54. K about the effect of perturbations in the accuracy of the K on the products of using that K in problem solving (K about the importance of the accuracy of a given bit of K).

aes 55. K about groupings of values for particular concepts which lead to identical effects in the use of those concepts in handling problems (K of the effective grainsize of domain variables).

aes 56. K of constraint graph relationships between other K.

APPENDIX B

GENERAL KNOWLEDGE AND DOMAIN-SPECIFIC KNOWLEDGE

Here is a chart of types of general and domain specific knowledge along with the same kind of annotation that was provided with the metaknowledge chart in Appendix A. Some of the metaknowledge in Appendix A was also domain specific, so this is not meant as a dichotomy with the types of knowledge presented. Again, the annotation in the first column is to be interpreted as follows:

A - The knowledge acquirer must know.

E - The knowledge engineer must know.

S - The system must know.

Lower case indicates that it is not necessary for them to know, although it would be nice if they did. An omitted annotation indicates that they need not know.

Who Need Know		Kind of General or Domain Specific Knowledge
aes	1.	K about planning in general (what is necessary for formulating plans in any domain for any problem).
aes	2.	K of goals.
aEs	3.	K of interrelationships among goals:
		a. Priorities.
		b. Which goals are subgoals of other goals.
		c. Which can be jointly satisfied by the same action or series of actions.
aES	4.	K of the durations of particular events and actions.
aEs	5.	K of what actions can occur in parallel (simultaneously).
AES	6.	K of what decisions must be made.
aEs	7.	K of the interrelationships between decisions which need to be made.
aES	8.	K of what decisions may be postponed.

aEs 9. K of what constitutes a good order for tackling the problem.

AES 10. K of what are the objects in the domain.

AES 11a. K of the basic concepts (primitives) in the domain.

aes 11b. (K that these are the basic concepts (primitives) in the domain).

aES 12. K of locations and relative positions in space and time.

AEs 13. K of the reasoning strategies used in the domain, e.g., the role of the following in solving problems in the domain of interest:

a. Analogy.

b. Decomposition and recombining.

c. Conjecturing.

d. Generalization.

e. Specialization.

f. Identification of limits.

g. Construction.

AES 14. K of the available actions to accomplish goals.

AES 15. K of constraints on possible actions.

aES 16. K of the effects of actions on the world.

aes 17. K of the interactions among objects in the domain.

AES 18. K of rules of thumb (and other shortcuts or aids to handling problems) used by experts in solving problems in the domain.

es 19. K of probabilistic facts (as opposed to uncertain knowledge).

aES 20. K of the preconditions of actions.

aEs 21. K of in what situations particular actions are appropriate.

aEs 22. K of side effects of actions.

aes 23. K of random processes in the domain.

aes 24. K of periodicities.

aes 25. K of feedback.

aes 26. K of controllability of the observables and the possibility for experimentation in the domain.

aes 27. K of the types of tasks to be performed in the domain, e.g.:

a. Classificatory.

b. Predictive (what-would-happen-if).

c. Plan synthesis.

d. Detection.

e. Manipulation.

f. Speech understanding.

g. Image interpretation.

h. Design.

i. Diagnosis.

aes 28. K of sizes and shapes.

aes 29. K of resources.

aes 30. K of states, abstractions of states, and constraints on states.

aes 31. K of schedules.

aes 32. K of time windows.

aes 33. K of plans.

aes 34. K of procedures.

aes 35. K of activities.

aes 36. K of situations.

aes 37. K of events.

aes 38. K of causality, or other connections between events and situations in the domain.

AES 39. K of relations and functions on objects in the domain.

aes 40. K of priorities.

APPENDIX C

THE SIMPLEX AND KARMARKAR ALGORITHMS

Linear programming techniques offer ways to handle constraints on continuous variables to obtain optimized solutions. The classical method for attacking these problems is the simplex algorithm developed by George Danzig in the 1940s (Strang, 1986). This algorithm starts out with a graphical representation of the problem, using all straight line functions, since the constraints are linear, and steps along the edges of the allowed region, from one corner to the adjacent one, moving between choices until the optimum one is found. The best choice is always a corner since the programming problem is linear. In complex problems, this stepping towards the solution is done in a multidimensional space (possibly even thousands of dimensions may be involved) and solutions may take hundreds of hours to compute.

The Karmarkar algorithm is a new approach to solving complex realistic problems (Strang, 1986). It is at least 50 times faster than the simplex algorithm on problems with thousands of constraints, and could become even better when fully developed. It has not been officially published yet, but the key idea is to begin with a sphere inscribed in the allowed solution space and to move to the point on the sphere where the desired quantity is optimized. Another sphere is drawn around that point, again with the requirement that it be entirely within the solution space and, hence, within the constraints of the original problem.

The relative advantage of Karmarkar's algorithm over the simplex algorithm grows as the problems get bigger, since the first few steps of Karmarkar's approach can do as much as thousands of simplex steps in a big problem. Various elaborations of Karmarkar's method using projective transformations remap the problem dynamically so that the solution ends up at the origin of the coordinate system. A variety of other tricks improve its run time efficiency.

The impetus to formulating the parameters of complex problems as linear programming problems may be intense once these methods are fully developed and readily available. This is another reason for acquiring knowledge of the constraint relations between continuous variables for the air strike planning domain.

Example: Suppose, for the sake of argument, that the only constraints on a choice of weapons are their weights and their cost, and that there are only two weapon types to consider, A and B (figure C-1). In two dimensions, the problem can be represented graphically as shown in figure C-1.

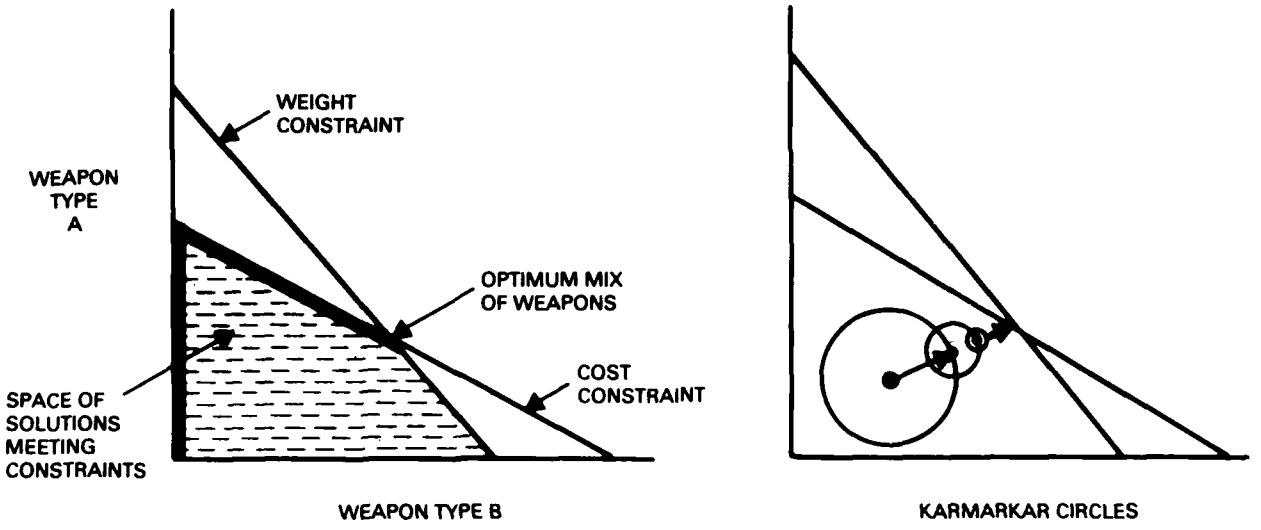


Figure C-1. Comparison of Karmarkar and Simplex methods.

APPENDIX D

REFERENCES KEYED TO SYSTEMS

References to works pertaining to the various systems, shells, and languages discussed in this report are given in an abbreviated form in table D-1. Full references are given in the reference section.

Table D-1. Short-form references for systems and tools.

System, Tool or Language	Short references
ACLS	Paterson and Niblett, 1982 (see also EXPERT-EASE).
ADVISE	Michalski and Baskin, 1983; Boulanger, 1983; Michalski, Baskin, Seyler, and Boulanger, 1984; Reinke, 1983; Rodewald, 1984.
AGE	Aiello, Bock, Nii, and White, 1981; Aiello and Nii, 1981; Nii and Aiello, 1979; Hayes-Roth, Waterman, and Lenat, 1983.
AQ11	Michalski and Chilausky, 1980a and 1980b; Chilausky, Jacobsen, and Michalski, 1976; Michalski, 1983.
ART	Clayton, 1984; Williams, 1984.
ATEST	Michalski and Baskin, 1983; Michalski, Baskin, Seyler, and Boulanger, 1983.
ATTENDING	Miller, 1984.
BABY	Rodewald, 1984.
BACON	Langley, Bradshaw, and Simon, 1983; Langley, Zytkow, Simon, and Bradshaw, 1986.
CASNET	Alty and Coombs, 1984; Kulikowski, 1983; Weiss and Kulikowski, 1981; Weiss, Kulikowski, Amarel, and Safir, 1978.
CHECK	Nguyen, Perkins, Laffey, and Pecora, 1985.
CLOT	Bennett and Engelmore, 1984.
CLUSTER/2	Michalski and Stepp, 1983.
COBWEB	Fisher, 1986.

Table D-1. Short-form references for systems and tools (continued).

System, Tool or Language	Short references
DUCK	McDermott, D., 1984.
EMYCIN	van Melle, Shortliffe, and Buchanan, 1981 and 1984; Hayes-Roth, Waterman, and Lenat, 1983.
ETS	Boose, 1984; Boose, 1985a and 1985b; Shaw, 1980, 1981, and 1984; Gaines and Shaw, 1981.
EXPERT	Kulikowski, 1983; Weiss and Kulikowski, 1979, 1981 and 1984; Weiss, Kern, and Kulikowski, 1980; Hayes-Roth, Waterman and Lenat, 1983.
EXPERT-EASE	Perrone, 1983.
FRL	Goldstein and Roberts, 1979.
GEM	Michalski and Baskin, 1983; Michalski, Baskin, Seyler, and Boulanger, 1984; Paterson, 1983.
GODDESS	Pearl, Leal, and Saleh, 1982.
HEARSAY II	Erman, London, and Fikas, 1981; Hayes-Roth, Waterman, and Lenat, 1983.
HPRL	Lanam, Letsinger, Rosenberg, Huyun, and Lemon, 1984; Rosenberg, 1983.
ID3	Quinlan, 1979, 1983a and b, 1986.
IPS	Rychener, 1981 and 1983.
KAS	Reboh, 1979 and 1981; Hayes-Roth, Waterman, and Lenat, 1983.
KEE	Kehler and Clemson, 1984; Kunz, Kehler, and Williams, 1984.
KLASSIC	Finin, unpublished.
KRL	Bobrow and Winograd, 1977.
KuBIC	Finin and Silverman, 1984; Silverman, 1984.
LEX	Michell, Utgoff, and Banerji, 1983.
LOOPS	Bobrow and Stefik, 1983; Stefik, Bobrow, Mittal, and Conway, 1983.

Table D-1. Short-form references for systems and tools (continued).

System, Tool or Language	Short references
MDX	Chandresekaran and Mittal, 1983; Mittal and Dym, 1985.
METADENDRAL	Lindsay, Buchanan, Feigenbaum, and Lederberg, 1980; Buchanan and Feigenbaum, 1978; Buchanan and Shortliffe, 1984.
MORE	Kahn, Nowlan, and McDermott, 1984 and 1985.
MRS	Genesereth, 1983a 1983b and 1984.
MYCIN	Buchanan and Shortliffe, 1984; Davis, Buchanan, and Shortliffe, 1977; Shortliffe, 1976.
NANOKLAUS	Haas and Hendrix, 1983.
ONCOCIN	Buchanan and Shortliffe, 1984; Langlotz and Shortliffe, 1983, Suwa, Scott, and Shortliffe, 1982; Shortliffe, 1981.
OPS	Forgy and McDermot, 1977.
OPS5	Brownston, Farrell, Kant, and Martin, 1985; Forgy, 1981; Hayes-Roth, Waterman, and Lenat, 1983.
OPS83	Forgy, 1984.
PLANT/CD	Boulanger, 1983.
PLANT/ds	Michalski, Davis, Bisht, and Sinclair, 1983; Michalski and Chilauski, 1980a and 1980b.
PRISM	Langley, 1983b and 1985; Ohlsson and Langley, 1986.
PROLOG	Robinson, 1979; Clocksin and Mellish, 1982; Bratko, 1986; Walker, McCord, Sowa, and Wilson, 1987.
PROSPECTOR	Duda, Gaschnig, and Hart, 1979; Duda and Reboh, 1984; Gaschnig, 1982; Hart and Duda, 1977.
PUFF	Aikins, Kunz, Shortliffe, and Fallat, 1983; Hayes-Roth, Waterman, and Lenat, 1983; Freiher, 1980.
RLL	Greiner and Lenat, 1980a and b; Hayes-Roth, Waterman, and Lenat, 1983.

Table D-1. Short-form references for systems and tools (continued).

System, Tool or Language	Short references
R1	McDermott, J., 1981, 1982a, and 1984; Rosenbloom, Laird, McDermott, Newell, and Orciuch, 1984.
ROGET	Bennett, 1984.
ROSIE	Hayes-Roth, Klahr, and Mostow, 1981; Hayes-Roth, Waterman, and Lenat, 1983; Fain et al., 1981.
RUMMAGE	Fisher, 1984.
SACON	Bennett and Engelmore, 1984; Buchanan and Shortliffe, 1984.
SAGE	Langley, 1982 and 1983a.
SALT	Marcus, McDermott, and Wang, 1985.
SEEK	Politakis, 1982; Politakis and Weiss, 1982 and 1984.
SEEK2	Ginsberg, Weiss, and Politakis, 1985.
STAGGER	Schlimmer and Granger, 1986.
TEIRESIAS	Davis, 1976, 1978 and 1979; Davis and Lenat, 1982.
UNITS	Stefik, 1979.
XCON	McDermott, J., 1982b and 1984.
XSEL	McDermott, J., 1982a.

END

FILMED

MARCH, 1988

DTIC