| NASA |

| NASA |

USL / DBMS NASA / RECON

WORKING PAPER SERIES

Report Number

DBMS.NASA/RECON-5

The USL/DBMS NASA/RECON Working Paper Series contains a collection of reports representing results of activities being conducted by the Computer Science Department of the University of Southwestern Louisiana pursuant to the specifications of National Aeronautics and Space Administration Contract Number NASW-3846. The work on this contract is being performed jointly by the University of Southwestern Louisiana and Southern University.

For more information, contact:

Wayne D. Dominick

Editor
USL/DBMS NASA/RECON Working Paper Series
Computer Science Department
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504
(318) 231-6308

| N A S A |

KNOWLEDGE BASED SYSTEMS:

A PRELIMINARY SURVEY

OF

SELECTED ISSUES AND TECHNIQUES

Srinu Kavi

Computer Science Department
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504

May 31, 1984

ABSTRACT

It is only recently that research in artificial intelligence (AI) is accomplishing practical results. Most of these results can be attributed to the design and use of expert systems (or Knowledge-Based Systems, KBS) - problem solving computer programs that can reach a level of performance comparable to that of a human expert in some specialized problem domain. But many computer systems designed to see images, hear sounds and recognize speech are still in fairly early stage of development.

In this report, a preliminary survey of recent work in KBSs is reported, explaining KBS concepts and issues and techniques used to construct KBS. Application considerations to construct KBSs and potential research areas in KBSs are identified. A case study (MYCIN) of a KBS is also provided.

TABLE OF CONTENTS

Within this report, certain survey sections are based heavily on state-of-the-art research reported on by other authors. Such section titles within the report are followed immediately by cited references indicating that credit for the contents of the entire section is due to the cited author(s).

KNOLEDGE BASED SYSTEMS:
A PRELIMINARY SURVEY
OF
SELECTED ISSUES AND TECHNIQUES

1. INTRODUCTION

It is only recently that artificial intelligence (AI) - that branch of computer science that attempts to have machines emulate intelligent behavior - is accomplishing practical results. Most of these results can be attributed to the design and use of expert systems (or Knowledge-Based Systems, KBS) - problem solving computer programs that can reach a level of performance comparable to that of a human expert in some specialized problem domain [Nau, 83].

This project report surveys recent work in KBS, explaining KBS concepts and issues.

2. KNOWLEDGE-BASED SYSTEMS (KBS)

It is necessary to distinguish, at the outset, between knowledge-based systems and other computer-based systems that contain or incorporate knowledge. Almost all computer programs

and systems contain knowledge of at least two kinds: knowledge about things and knowledge about what to do with things - that is, how to manipulate or transform them. Davis [Davis, 77] defines the KBS in the following way: 'A knowledge-based system is one in which knowledge is collected in one or more compartments (called knowledge sources) and is of the kind that facilitates problem solving (reasoning) in a single, well-defined problem domain.'

Problems are solved by applying the kind of reasoning that is used by a practitioner in the domain in which KBS is applied. Unlike generalized problem solving system, KBS must accumulate large amounts of knowledge in a specific domain and rely on domain-specific problem solving techniques that can be developed to a high level of expertise [Davis, 77].

In building such systems, researchers have found that amassing a large amount of data rather than sophisticated reasoning techniques is responsible for most of the power of the system. Such expert systems, previously limited to academic research projects, are beginning to enter the software market place [Gevarter, 83]. Some of the application areas where KBS are used are:

(1) Medical diagnosis.

(2) Mineral exploration.

(3) Oil-well log interpretation.

(4) Chemical and biological synthesis.

(5) Military threat assessment.

(6) Planning and scheduling.

(7) Signal interpretation.

(8) Air-traffic control.

(9) VLSI design.

(10) Equipment fault diagnosis.

(11) Speech understanding.

(12) Space defence.

(13) KB access and management.

Table 2-1, taken from [Nau, 83], lists few of the existing systems developed for some problem areas. More extensive lists can be found in [Gevarter, 83].

In addition to these systems, some compiler languages are being developed to provide tools for creating knowledge-based systems: AGE [Nii, 79], ARS [Stallman, 77], KRL [Bobrow, 77], OPS [Forgy, 77].

Speech-understanding systems developed at CMU (among other places) are not included in this survey [Stefik, 77; Reddy, 73; Reddy, 75]. This is because they are more complex and more of research prototypes, yet many of the techniques they embody are incorporated in systems discussed in this report.

Table 2-1 SOME EXISTING EXPERT SYSTEMS [Nau, 83]

SYSTEM	EXPERTISE
AQ1	Diagnosis of Plant Diseases
CASNET	Medical Consulting
DENDRAL	Hypothesizing Molecular Structure from Mass Spectrograms
DIPMETER ADVISOR	Oil Exploration
EL	Analyzing Electrical Circuits
INTERNIST	Medical Consulting
KMS	Medical Consulting
MACSYMA	Mathematical Formula Manipulation
MDX	Medical Consulting
MOLGEN	Planning DNA Experiments
MYCIN	Medcial Consulting
PROSPECTOR	Mineral Exploration
PUFF	Medical Consulting
R1	Computer Configuration

Before we go to next section, I think it's appropriate to "define" knowledge. The following is taken from [Wiederhold, 84]. Widerhold observes that:

(1) Knowledge considers general aspects of data.

(2) Knowledge is significantly smaller than data.

(3) Knowledge does not vary rapidly (compared to data).

Widerhold gives some simple examples to illustrate the difference between knowledge and data:

Mr. Lee's age is 43 years	- Data
Middle-age is 35 to 50	- Knowledge
People of middle-age are careful	- Knowledge
Mr. Lee has never had a traffic accident	- Data

2.1 A Hypothetical KBS

The following is a brief description and example of a simple, hypothetical KBS application that illustrates most of the capabilities of a KB system. This is mostly based on [Davis, 77].

The hypothetical system is an automotive service consultant whose primary purpose is to help ensure the best service at the least cost for automobiles brought to a service agency. The system is to be used both by the service representative, who is the primary interface between the customers and the company, and by the mechanics who work on the cars. Although the reasoning skills required by such a KB system are relatively simple, the amount of knowledge required (exclusive of the normal data base) is large because of the number and variety of automotive subsystems involved and the high degree of their interdependence.

The data base of the system would contain a description of each car serviced at the agency. The description would contain the configuration of the vehicle's engine, running gear, and optional and special equipment as well as the year of manufacture, model, color, service history, etc.

The system would contain two basic kinds of knowledge:

(1) Generic knowledge about the major elements of the vehicle (such as its cooling, ignition, and fuel systems).

(2) Knowledge about the model and its specific options.

Diagnostic knowledge would be necessary to relate symptoms to causes and to relate specific knowledge to generic knowledge. For instance, poor engine performance could result from mechanical problems (valves, rings) or from related systems (cooling, ignition, or carburetion).

The intent of the system is to recommend:

(1) The least expansive tests that will produce the greatest amount of specific information about the cause of the problem.

(2) The least expensive repairs for the problem.

(3) The best post-repair testing to verify that the problem is solved.

The following is a hypothetical scenario of such a system in operation. The users are the service representative and the mechanic(s) performing the test and repair work. The user

language is a stylized English designed to convey a maximum amount of information between the user and the system with a minimum amount of effort.

System: Tuesday, May 3, 1977, 08:32:44

Service rep: Next customer

System: Enter owner's name or vehicle ID number

Service rep: 1743 HBAQ 8197

System: Owner is S.B. Franklin, car is 1974 Ford Gran Torino 4-dr sedan with 351-2V engine, A/C, P/S, P/B and CA smog controls. Last service - April 4, 1977 at 28,412 mi R&R water pump and lower radiator hose, what is present mileage?

Service rep: 29285

System: Please inform owner that the following regular maintenance services are due:

Change oil and filter - last changed on Jan 13, 1977 at 24,842 mi.

May we be of other service?

Service rep: Battery not keeping charge, barely able to start car this a.m.

System: Any other symptoms or problems?

Service rep: No

System: Check belt tension on alternator - most likely cause.

Service rep: Why?

System: Last service required removal of fan and belts, very likely that tension was not properly set on reinstallation.

Service rep: More (meaning more things to be checked or tested)

System: Check fluid level in battery - next most likely cause - overcharged battery. If level is low, check alternator output and voltage regulator.

Service rep: Replace battery?

System: No, battery is 6 mos old and should take recharge.

Service rep: Thank you. (Signal to system that session is ended for now)

System: Car assigned number 287 in control system. First test to be performed by J.K. Uzzi in Bay 17.

After the car is moved to the mechanic's work area and the prescribed tests are performed, the following interchange takes place between the mechanic and the system:

Mechanic: JKU 17, 287 (indicating mechanic, bay and car being serviced)

System: 1974 Gran Torino, test results please.

Mechanic: Belt tension OK, battery level OK, charge low, alternator, VR output OK. Problem not found.

System: Test for above-normal current flow with ignition off. (Mechanic attaches ammeter and takes reading).

Mechanic: Reading above normal.

System: Check for faulty switch in hood light, trunk light, and courtesy lights.

Mechanic finds faulty switch in trunk light such that light never turns off.

Mechanic: Fault trunk light switch R&R. Current reading normal.

What specific knowledge must such a system incorporate in order to diagnose and suggest remedies? The knowledge must come from experts who have acquired and demonstrated diagnostic skills. For the system to have suggested checking the belt tension of the alternator, it would have to know that the earlier removal of the belt could be related to the present problem, that the severity of the problem would depend on how poorly the tension was adjusted, and that one month and about 900 miles before appearance of symptoms (battery failure) is not unreasonable. Since it is a highly probable cause and the easiest to test, it ranks as the first suggestion. By requesting more information, the service representative can tell the owner what else may be required and what will not likely be required, such as a new battery.

Even though this is an hypothetical system, functionally it is similar to many successful KB systems existing today. Of course, those systems solve more difficult problems in the sense that the reasoning chains may be longer. The knowledge, however,

is of a similar variety, and the interactive discourse has the same flavor of naturalness.

2.2 KBS Components

A knowledge-based system is one that supports practitioners in a specific knowledge domain by incorporating the knowledge acquired from experts in that domain and applying it, in combination with certain reasoning skills, to the solution of problems posed by the practitioners. In other words, a KBS functions as an intelligent assistant - a substitute for the expert human consultant who may be needed but unavailable. A KBS may produce solutions or explanations that are more thorough than those produced by a human expert, and may produce them more rapidly [Gevarter, 83]. However, the human has imagination and the capacity for innovation which even the most expert KBS does not have.

A KBS is composed of three components or modules:

(1) An interface.

(2) A cognitive engine.

(3) A knowledge base (Figure 2-1).

The knowledge base - the passive element in the system - contains the knowledge sources and fact files.

The cognitive engine drives the system. It performs the system's problem solving (inference-making or reasoning) operations. It applies the knowledge in the knowledge sources and uses the fact files in the knowledge base to answer questions or solve the problem posed by the user.

The interface provides interactive communication between the user and the system. It allows for the acquisition of data in a variety of forms - a real-time signal, a file of observations, data provided by the user, etc, and for the addition or modification of knowledge in the knowledge base.

As already mentioned above, a KBS acts as a special-purpose "intelligent agent" on behalf of its user; it is not a general-purpose problem solver. It provides supportive knowledge in a well-defined, clearly bounded problem domain. No present-day KBS is intended for use by a casual, inexpert user.

| N A S A |

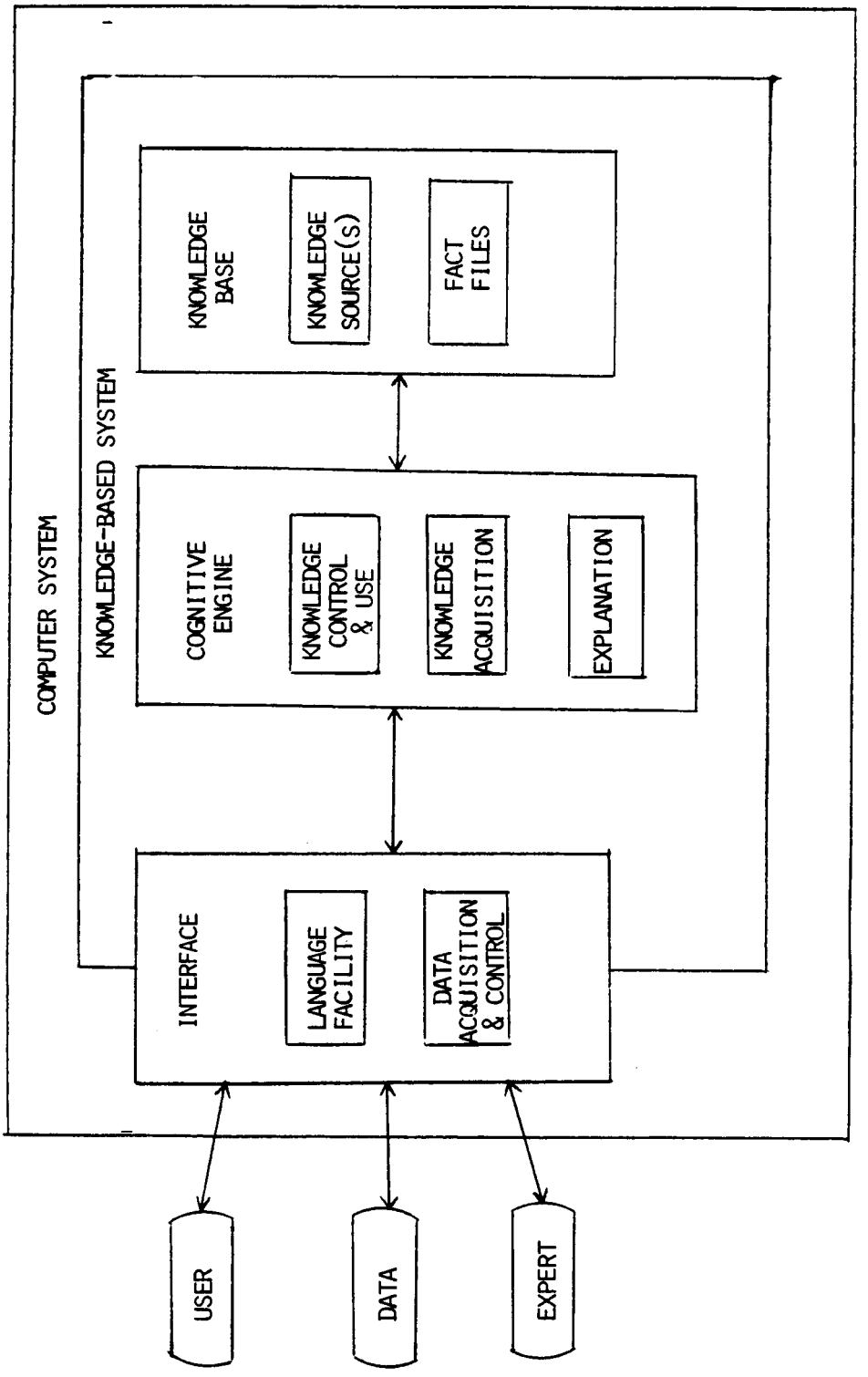


FIGURE 2.1 KBS ELEMENTS AND THEIR RELATIONSHIP

2.3 Knowledge Base

As already mentioned, the Knowledge Base (KB) is a passive element of KBS. Though passive, the KB determines the performance and utility of the KBS, because the Cognitive Engine (CE) (see Section 2.4) depends on the knowledge in the KB. In this section, the characteristics of the KB that are common to many of the KB systems (that I studied) are described.

The KB of a KBS may contain both Knowledge Sources (KSs) and fact files. The fact files that are contained in a KB are equivalent to a data base in that they contain attribute values and the equivalent type of information that may be required for the complete solution of a problem. A collection of fact files without a Knowledge Source is not a KB. A Management Information System constructed from a conventional data-management system is not a KBS because it does not have reasoning or inferencing capability.

A KS contains what is usually called knowledge. Multiple KSs are usually necessary when there are multiple "levels" of knowledge, such as problem-specific knowledge and knowledge (often called Meta-knowledge) about how the CE can best use the problem-specific knowledge. Since the two kinds of knowledge are

used for different purposes, it is reasonable to keep them in different KSs. It is also often true that more than one kind of problem-specific knowledge is acquired from different experts and that there is no efficient single method for representing all of the knowledge. Since different representations are needed, separation into separate KSs is logical.

The knowledge of the KSs is usually of the following types [Shortliffe, 76; Davis, at.al, 77; Nau, 83]:

(1) Methods for specifying cause-effect relationships, implications, or inferences, using production rules, predicate-calculus expressions, or other logical representations, depending on the richness of the relationship to be represented.

(2) Plans of action for how one would achieve an end result in the world external to the model that the system represents. For instance, such a procedure may describe how to transform one chemical compound into another for a chemical-synthesis system, or how to purify an intermediate compound.

(3) Declaratives that identify objects within the modeled domain and distinguish them from objects that are not

within the domain. These declaratives may describe properties of objects, relationships among objects, definitions of terms or constructs, schemata that identify the legal relationshlips or transformations applicable to the domain, etc.

(4) Meta-properties, which are a higher level of abstraction about the domain and the solution space and methods. They take the form of meta-rules - that is, rules about using the knowledge in types (1), (2), and (3) above. They provide a means for determining and assuring the consistency, coherency, and reliability of intermediate results and steps as well as the final solution and answers. They may also restrict the solution space in various ways (such as pruning and ordering a "move" tree) that markedly improves the effeciency of the system.

(5) Advice (sometimes called heuristics) that is similar to meta-properties in intent, but that does not carry the same strength of influence. Advice may be a hint to the CE as to what knowledge is best to use next or how to escape from a possible blind alley or what is the most likely transformation that will yield a useful result.

There are a variety of techniques that have been used to represent KSs with these characteristics [Feigenbaum, 81]. They will be described in detail in Chapter 3.

2.4 Cognitive Engine

The Cognitive Engine (CE) combines and organizes the contents of the KB in inference or search structures in order to perform plausible or common-sense reasoning about the domain as it applies to the problem posed by the user. The intent of the CE is to focus the effort as narrowly as possible on the problem at hand.

The CE provides the central control of the KBS. How the CE does this affects both the power and the performance of the system, but is not the sole determinant. A KBS's ability to solve a particular problem depends on:

(1) How many paths there are to a solution.

(2) The ability of the Cognitive Engine to reduce the number to a minimum.

(3) The knowledge in the Knowledge Base.

(4) What information is available within the problem

statement.

Therefore, although the Cognitive Engine is in command and acts as the driving element, the path to a solution, and the criteria for when to accept a solution or abort a particular path are highly dependent on the content of the KB and the problem data.

To qualify as a KBS, a system should possess the potential for explaining its actions and reasoning processes with respect to an interaction with the user or to a solution it produces. This is another function of the Cognitive Engine. Explanations are given in terms of the content of the KB, the problem data and prior interactions with the user and are related only to past activity; the system cannot explain how it might deal with a hypothetical case or how it will continue in solving a present problem.

The CE must also provide the mechanisms that facilitate the acquisition of new knowledge, the modification of existing knowledge, and deleting erroneous or useless knowledge - all of which are done in cooperation with an expert. A KBS does not generally permit its users to make permanent additions or changes to the Knowledge Base.

The knowledge contained in the CE may be general or

meta-knowledge about how to reason (infer or search) as well as domain-specific problem-solving knowledge. The ultimate decision about what kind and what level of knowledge to incorporate in the CE depends on the intent of the system and the complexity of the domain, as well as on considerations about performance, efficiency, growth, and so on.

In summary, the CE is the controlling, active element of the KBS that directs the problem-solving activities, explains the system's behavior to its users upon request, and manages the Knowledge Base.

Some of the techniques of CE are described in Chapter 3.

2.5 Interface

The interface is the communication port between the system and the external world. As such, it provides three functions:

(1) Interaction with the user (i.e., accepting input and returning results, explanations, or other output often in English or a stylized natural language of the domain).

(2) Addition of knowledge to the KB by a domain expert.

(3) Acquisition by the KB of problem data.

These functions are explained in more detail below:

(1) The User Interface should accommodate the jargon specific to the domain of the KBS and may permit a natural language. It provides the necessary facilities for the user as a poser of problems and a consumer of results (answers, explanations). It is not the port through which expert knowledge is entered into the system, nor is it intended to support casual, inexpert users.

(2) The Knowledge-Acquisition (Expert) Interface is used by a domain expert (who has gained some feeling for the system) as the provider of knowledge for the KSs. Associated with the Knowledge-Acquisition Interface is some means of verifying the incoming knowledge, sometimes limited to syntax checking, but often including tests for coherence and consistency with prior knowledge both in the KSs and the CE [Shortliffe, 76]. It is possible that the KA Interface and the User interface use some system components in common, but they are considered

logically separate.

(3) The Data Interface is more conventional than the other two. It is similar to that of most other interactive computer systems, in that it incorporates:

(a) Facilities for user input of data and responses to the system's queries.

(b) The mechanism for locating and accessing files and data bases.

Many of the functions necessary to provide the Data Interface may be drawn directly from the computer-system environment within which the KBS functions.

Separation of KBS Elements

The separation of the elements of a KBS is a necessary condition for including a system in that category, since it permits the changing of the domain of application by extending, expanding, or substituting another KB independently of the CE. One example: EMYCIN (for Empty MYCIN) is the CE of MYCIN [Davis, et.al, 77] (also see Appendix A), to which several different KBs have been experimentally attached for solving different classes of problems. Still there is no general theory of knowledge-based systems complete enough to permit the facility of substituting a

KB from a different domain as a means of creating a new KBS for that domain.

Even though it was mentioned that a KBS must have a CE or a KB, in most of the systems they are not so easily identifiable.

2.6 Summary

In summary, to qualify as a KBS, a system must:

(1) Be externally invoked by an expert in the domain of applicability.

(2) Have an identifiable CE that reasons plausibly using the KB and whose solution path is controlled by the content of the KB and problem data.

(3) Have the potential for explaining its behavior.

(4) Have an identifiable KB that contains expert domain-secific knowledge (this is the most critical aspect of a KBS).

(5) Be organized and structured so that its KB can be expanded and extended and the system's performance improved.

3. TECHNIQUES USED TO CONSTRUCT KBS

Artificial Intelligence (AI) techniques are widely used in KBS. In addition to AI, several other computer science areas have developed techniques that are used in the construction of KBS. A partial list of the major contributions are summarized in Figure 3-1.

The following subsections discuss KBS technologies grouped according to function. Section 3.1 describes the methods used to represent the knowledge contained in the Knowledge Sources (KS). Section 3.2 describes the methods used to model and represent the work-space - the dynamic state of a system during its problem-solving activity. Section 3.3 describes techniques that are used to construct the Cognitive Engine (CE). Section 3.4 describes the techniques used to build the interface between the KBS and the user.

Table 3-1 ORIGINS OF KBS TECHNIQUES
(Based on [Hayes-Roth, 83])

ARTIFICIAL INTELLIGENCE (AI)

- Hueristic Search
- Inference and Deduction
- Pattern Matching
- Knowledge Representation and Acquisition
- System Organization

LANGUAGE PROCESSING

- Parsing and Understanding
- Question and Response Generation
- Knowledge Representation and Acquisition

THEORY OF PROGRAMMING LANGUAGES

- Formal Theory of Computational Power
- Control Structures
- Data Structures
- System Organization
- Parsing

MODELING AND SIMULATION

- Representation of Knowledge
- Control Structures
- Calculation of Approximations

DATA BASE MANAGEMENT

- Information Retrieval
- Updating
- File Organization

3.1 Knowledge Representation

In contrast to conventional database systems, KB systems require a knowledge base with diverse kinds of knowledge - knowledge about objects, about processes, and hard-to-represent common sense knowledge about goals, motivation, causality, time, actions, etc. Attempts to represent this breadth of knowledge raise many questions [McCalla, et.al, 83]:

(1) How do we structure the explicit knowledge in a knowledge base?

(2) How do we encode rules for manipulating a knowledge base's explicit knowledge to infer knowledge contained implicitly within the knowledge base?

(3) When do we undertake and how do we control such inferences?

(4) How do we formally specify the semantics of a knowledge base?

(5) How do we deal with incomplete knowledge?

(6) How do we extract the knowledge of an expert to initially "stock" the knowledge base?

(7) How do we automatically acquire new knowledge as time goes on so that the knowledge base can be kept current?

In this section we describe some of the techniques used to represent such kinds of knowledge.

3.1.1 Knowledge Representation Forms [Feigenbaum, 81]

A knowledge source may assume several different forms of representation through a KBS. The domain expert imparts new knowledge to the knowledge acquisition mechanism in the external form. The acquisition mechanism transforms or compiles the external representation into the physical form and merges the new knowledge into the appropriate KS. The physical form is a data structure such as a matrix, list, or n-tuple, or a procedural representation, or some combination of these forms. When another component of the system (such as the CE) accesses the KS, the logical form is used at the interface. The logical form defines the set of questions that can be answered immediately by the KS. Finally, knowledge is transformed back into the external form when the system provides explanations to the user. Figure 3-2 summarizes the transformations of knowledge representations throughout a KBS.

| N A S A |

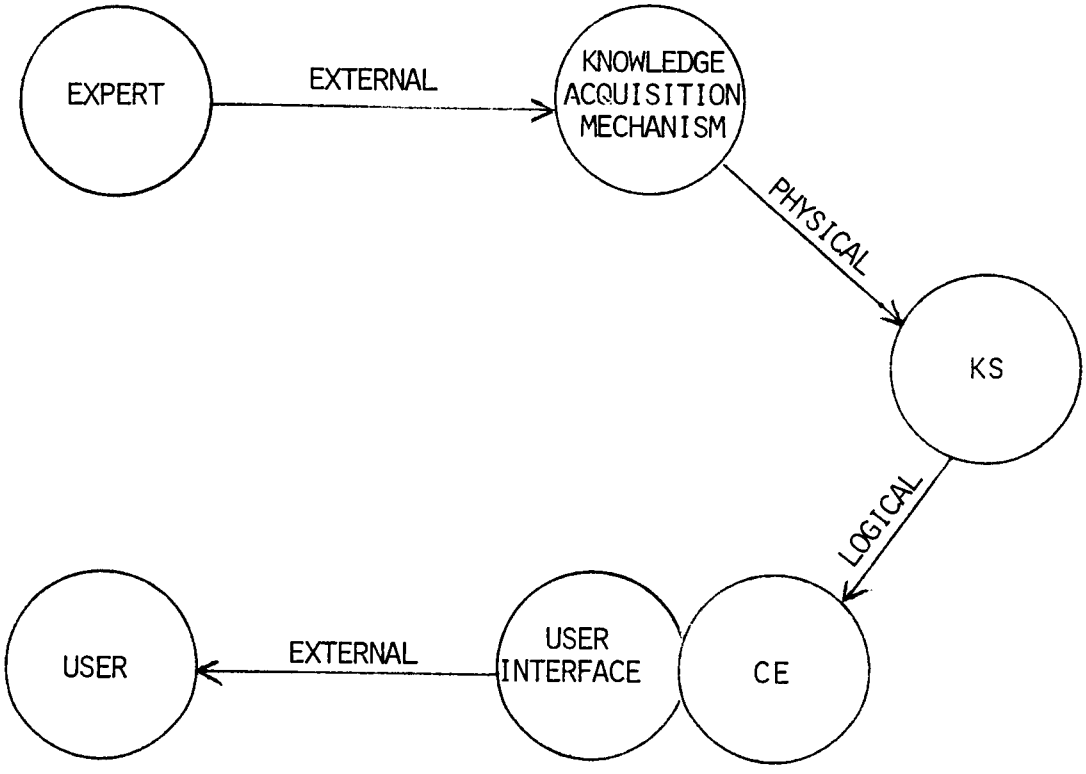


FIGURE 3.2 KNOWLEDGE REPRESENTATION FORMS

There are two important terms/concepts in knowledge representation [Feigenbaum, 81].

Knowledge Chunks. Both the external and logical knowledge representation format are partially characterized by the term chunk size. A knowledge chunk is a primitive unit in the representation - that is, in a KS that contains the definitions of several interrelated terms, the definition of a single term is a "chunk". The knowledge chunk is the unit by which the expert augments (or modifies) the KS. The simplest action that can be taken by the CE is to apply or use a single chunk. Chunk size of knowledge is an important consideration to KBS technology for three reasons:

(1) It determines the level at which the expert can instruct the system. If the chunk size is either too large or too small, the expert is forced into an unnatural mode of expressing his knowledge.

(2) It in part determines the acceptability of the system's explanation mechanism. Since the knowledge chunks used to derive and support the system's conclusions form the essential part of explanations, acceptability is enhanced when the chunks are the same

- size or level of detail used by one worker in the application field describing results to another worker in the same field.

(3) It determines the kinds and efficiency of reasoning techniques to be used in the KBS. Larger chunk sizes generally permit shorter lines of reasoning. For that reason, they are more likely to lead to a correct conclusion when inexact but plausible inference techniques are used.

Credibility Factors [Shortliffe, 76]. All of the knowledge in a KS need not be true in an axiomatic sense. Much of the content of a KS may be "rules of thumb" and working hypotheses. This raises the issue of how a system is to use knowledge of this sort to produce acceptable results. In many KB systems, the chunks in the KS are rated as to their credibility by the experts who entered them into the system. This rating is then available to the CE as a guide in the reasoning process.

There are at least four possible meanings or interpretations of credibility factors:

(1) A probability: the fraction of the time the chunk is true.

(2) - Strength of belief: how certain is the expert that the chunk is always true?

(3) Acceptability: is this a preferred method or fact to workers in the field?

(4) Relevence: what is the probability that use of this chunk will ultimalely lead to a completed chain of reasoning that solves the problem at hand?

It is essential that the kind of credibility factor that is to be used be stated and agreed upon by the expert who instructs the system and by the programmer who builds the CE.

Additional references include the following: [Tversky, 72; Zadeh 75a; Zadeh 75b; Goguen, 68].

3.1.2 Methods of Representing KS [Feigengaum, 81]

Some of the methods of representing a knowledge source:

(1) --Finite-state machine.

(2) Programs.

(3) Predicate calculus.

(4) - Production rules.

(5) Semantic networks.

(6) Frames.

(7) Conditional probabilities.

In this report I will describe only production rules, semantic networks and frames. The other methods will be discussed in a subsequent report.

3.1.3 Production Rules [Feigenbaum, 81]

Production rules have been used as the principal method of representing knowledge in many (if not most) of the highly successful KB systems - for example, MYCIN and DENDRAL.

A production rule is a specification of a conditional action. It consists of a left hand side (LHS), also called the condition or the antecedent, which describes a situation, and a right hand side (RHS), also called the action or consequence, which describes something that may legally be done in a situation described by the LHS.

For example, in "If you are outdoors and it is raining, then open an umbrella". The conditions are 'being outdoors' and

'rain'. The action is to open an umbrella.

Production system is a three-component entity:

(1) A collection of production rules.

(2) A workspace.

(3) A control mechanism.

The production rules are represented by some agreed-upon syntax, by means of which the LHS and RHS are built up from a set of primitives and symbols that correspond to objects, functions, and predicates in a domain. The workspace contains the total description of the system's current state or situation. The LHS of a rule describes, or is matched against, the contents of the workspace. If a production is applied, i.e., its LHS matches and its RHS is executed, then the RHS actions modify the workspace.

The control mechanism has two parts. The first part builds the conflict set - the set of all production rules whose LHSs are satisfied. If the conflict set is empty, then processing is terminated, and the result is the contents of the workspace. However, if the conflict set is not empty, then the conflict-resolution strategy selects one member of the conflict set and the RHS of the selected production rule is executed.

Several conflict-resolution strategies have been used or proposed.

Rule order - There is a complete ordering of all production rules. The rule in the conflict set that is highest in the ordering is chosen.

Rule precedence - A precedence network determines an ordering.

Generality order - The most specific rule is chosen.

Data order - Elements of the workspace are ordered. The rule chosen is the one whose LHS references the highest-ranking workspace element(s).

Regency order - Execute the rule in the conflict set that was most (least) recently executed, or the rule in the conflict set whose LHS references the most (least) recently referenced element(s).

Non-deterministic - Execute every rule in the conflict set as if it were the only member. Computation stops when any path terminates.

3.1.4 An Example

The example is a production system that assists the service manager and mechanics in an automobile repair agency (refer back to Section 2.1). The scenario for using this system is the arrival of a customer at the agency. He reports the symptoms and problems to the service representative, who then enters this information into the system. The system has at its disposal a data base of past problems, repairs and services performed on the vehicle, and a KS of production rules that describe cause-and-effect relationships among the performance characteristics and measurable attributes of an automobile. Using the reported information, the past-history data base, and the KS, a diagnostic and repair plan is formulated and implemented.

Figure 3-3 gives a few of the production rules that might be present in such a system. Each rule is named; however, the rule names are used only for convenience. The format of the rule is:

IF lhs1 C1 lhs2 .... Cn-1 lhsn
THEN rhs1[p1] K1 rhs2[p2] .... Km-1 rhsm[pm]

R1 IF fan belt tension is low
THEN alternator output will be low [.5] and engine will overheat [.2]

R2 IF alternator output is low
THEN battery charge will be low [.7]

R3 IF battery is low
THEN car will be difficult to start [.5]

R4 IF automatic choke malfunctions OR automatic choke needs adjustment
THEN car will be difficult to start [.8]

R5 IF battery is out of warranty
THEN battery charge may be low [.9]

R6 IF coolant is lost OR coolant system pressure cannot be maintained
THEN engine will overheat [.7]

R7 IF there is a high resistance short AND fuse is not blown
THEN battery charge will be low [.8]

R8 IF battery fluid is low
THEN battery will boil off fluid [.3]

R9 IF battery fluid is low
THEN battery charge will be low [.4]

Figure 3-3 PRODUCTION RULES FOR AUTOMOTIVE SYSTEM KS

where the C1 and K1 are the connectives AND or OR. The LHS is evertything between the keyworkds IF and THEN, and the RHS is everything following the THEN. Each lhs(i) is an observable or measurable condition predicate, e.g., that the tension of the fan belt is low or the engine is overheating. Each rhs(i)[p(i)] is a condition, rhs(i), that will follow with certainty or probability, p(i). Thus rule R1 says that, if the tension of the fan belt is low, then there are two possible consequences:

(1) That about one-half of the time the output of the alternator will be low.

(2) About one-fifth of the time the engine will overheat.

The other production rules, R2-R9, are interpreted in a similar manner.

A fact file in the system is shown in Figure 3-4. The information included for each observation or measure is the agent from whom to gather data and the relative difficulty (or cost) of gathering the data.

OBSERVATIONS	AGENT	DIFFICULTY
Alternate Output Level	Mech	4
Battery Charge Level	Mech	3
Battery Fluid Level	SrvR	2
Choke Adjustment	Mech	5
Choke Function	Mech	5
Coolant Level	SrvR	2
Coolant System Pressure	Mech	5
Difficulty to Start	Cust	1
Engine Temperature	Cust	1
Fan Belt Tension	Mech	3
Fuse Condition	SrvR	2
Short in Electric System	Mech	8
Voltage Regulator Level	Mech	4
Warranties	Data Base	0

Figure 3-4 DATA GATHERING PROCEDURE FACT FILE

There are four possible agents for data gathering:

(1) The customer (Cust).

(2) The historical data base

(3) Inspection by the service manager (SrvR).

(4) Measurement by the mechanic (Mech).

The difficulty information will be combined with the confidence factors in the production rules to formulate the most cost-effective and timely plan for the needed diagnostics and repairs.

Assume that a customer arrives at the agency with the vague complaint that his car is hard to start. The service manager enters this information, including appropriate customer and vehicle identification. The system then grows a structure similar to that shown in Figure 3-5. The boxes are labeled with observable or measurable symptoms and are connected by arrows labeled with the names of the production rule they represent. To the far right of each of the unknown values (e.g., the box labels, such as battery-fluid level), the associated agent and relative difficulty are listed. At this point, the system would check the data base for information about the battery's warranty. If nothing decisive was found, then the customer would be asked whether the car was running hot, and the service manager would

continue to make on-the-spot observations. Diagnostic procedures for causes not ruled out by the procedure to date would then be placed on an ordered schedule for a mechanic.

The ordering would be based upon:

(1) Cost effectiveness - a function of test difficulty, estimated probability of being necessary, and ability to eliminate other tests.

(2) Availability of resources - speciality mechanism and test equipment.

| N A S A |

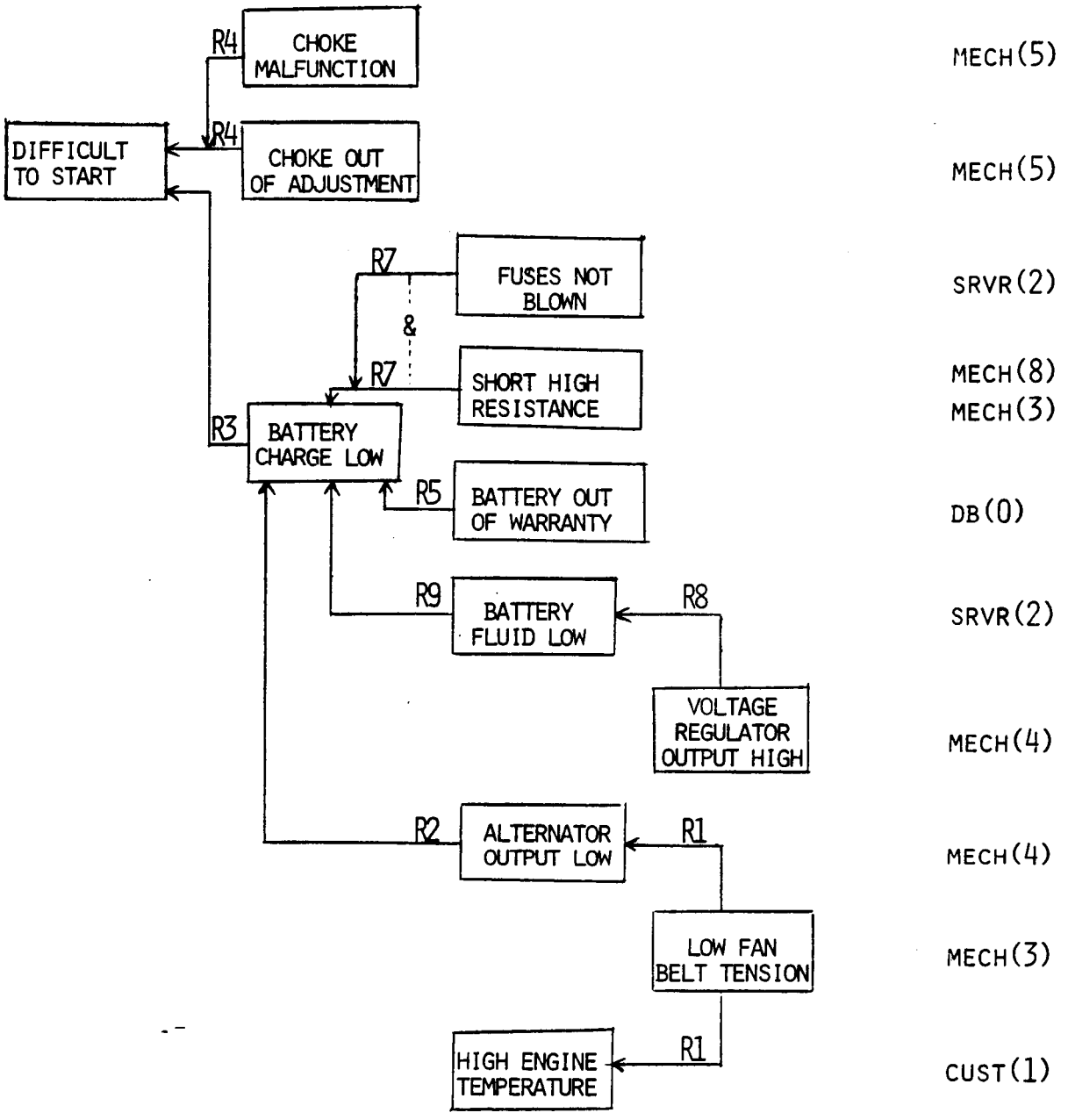


FIGURE 3.5 EXAMPLE FLOW IN AUTO DIAGNOSTIC SYSTEM

The structure shown in Figure 3-5 was grown by an algorithm called back chaining. A condition - in this case, "difficult to start" - is taken as a given, and the goal of the system is to find the cause(s). Note that the production rules state causes, then effects. Thus, the rules are used as if the knowledge possessed a kind of symmetry. The back-chaining algorithm is:

(1) Find all rules that have the initial or derived condition as their consequence (in this case R3 and R4).

(2) Call the antecedents (LHSs) of these rules' derived conditions.

(3) Repeat steps (1) and (2), and terminate when no more can be done.

Figure 3-6 graphically shows the kind of structure grown for each kind of rule format. In each example in the figure, c1 is the initial or a derived condition.

Rule E1 is the simplest; a1 is added to the set of derived conditions. Rule E2 states that if a1 is the case, then both c1 and c2 ought to follow. Thus, a1 is a derived condition, and c2 may or may not be considered a derived condition, depending upon

the particular strategy used by the system. Rule E3 is equivalent to two independent rules "IF a1 THEN C1" and "IF a1 THEN c2". Therefore, a1 is added to the set of derived conditions, and c2 part is ignored.

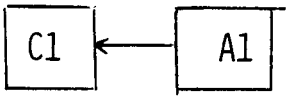
The example and discussion is some what simplistic, because there might be some problems which we did not consider. For example, suppose that rule R8 (in Figure 3-3) had been written more accurately as the two rules:

R8(1) IF voltage regulator output is high
THEN the battery will overcharge.

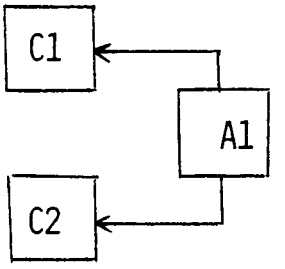
R8(2) IF battery is overcharged
THEN battery will boil off fluid.

| N A S A |

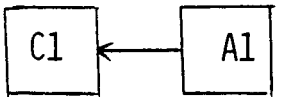
E1 IF A1 THEN C1



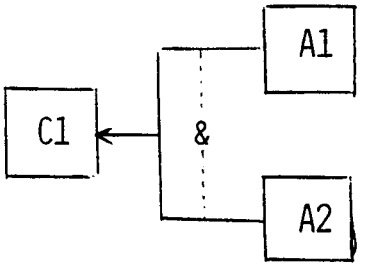
E2 IF A1 THEN C1 AND C2



E3 IF A1 THEN C1 OR C2



E4 IF A1 AND A2 THEN C1



E5 IF A1 OR A2 THEN C1

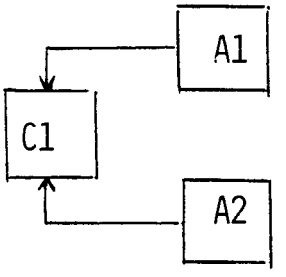


FIGURE 3.6 BACK CHAINING

With these new rules, a fragment of the structure shown in Figure 3-5 would be replaced by that shown in Figure 3-7. Now, the interesting conclusion is that a high battery charge implies a low battery charge. This is an apparent contradiction, since both conditions cannot hold at the same time. This kind of situation can often arise in unpredicted ways if the system contains many rules - more than a few dozen. The charge of the battery will oscillate between high and low as the battery fluid is replaced and boils off, respectively. So, in a sense, there is a missing rule of the form that adding fluid to a battery whose charge and fluid levels are low will probably allow the battery to return to normal conditions. However, to handle this kind of situation in general, it is necessary that the control mechanism or CE have some knowledge about how to proceed when faced with apparent conflicts and contradictions. One virtue of production systems is that ad hoc knowledge may be relatively easily incorporated in the system to handle this.

Additional relevant references include the following: [Davis, et.al, 77; Shortliffe, 76; Shortliffe 75; Buchanan, 76; Feigenbaum, 71; Barnett, 75; Collins, 76; Forgy, 76].

| N A S A |

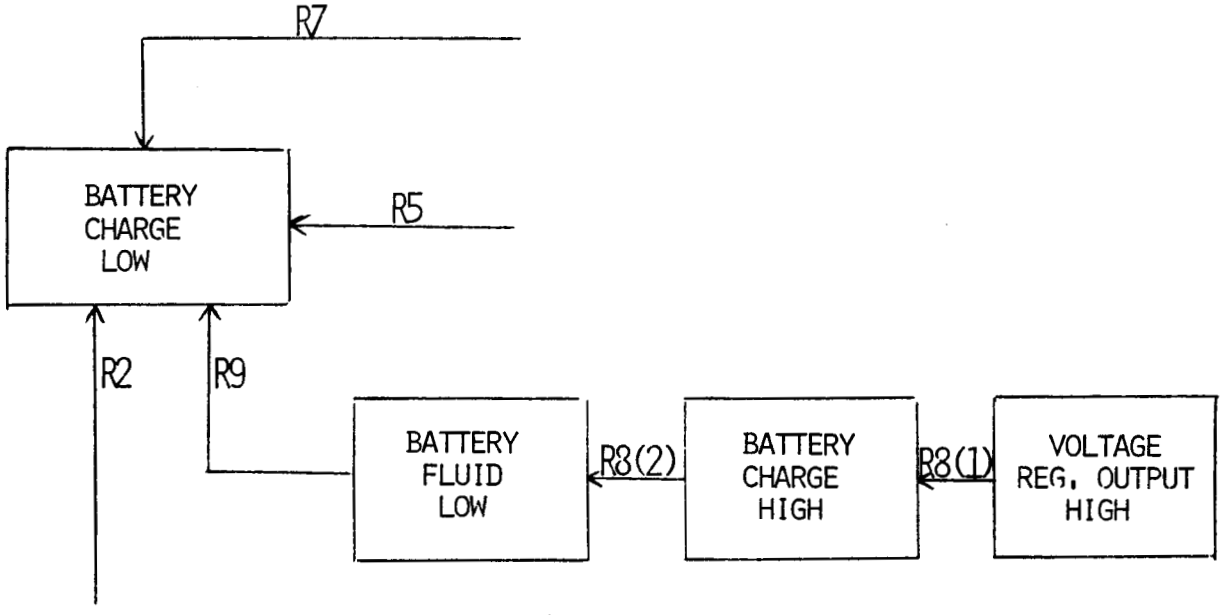


FIGURE 3.7 FRAGMENT OF GRAPH STRUCTURE

3.1.5 Semantic Networks [Nau, 83; Feigenbaum, 81]

A semantic network is a method of representing declarative knowledge about the relations among entities. The major application has been to include non-syntactic knowledge (e.g., semantics) in natural-language understanding systems. Because of their inherent generality and naturalness, semantic network have been used to represent highly interrelated information that cannot be properly processed by standard data base management techniques.

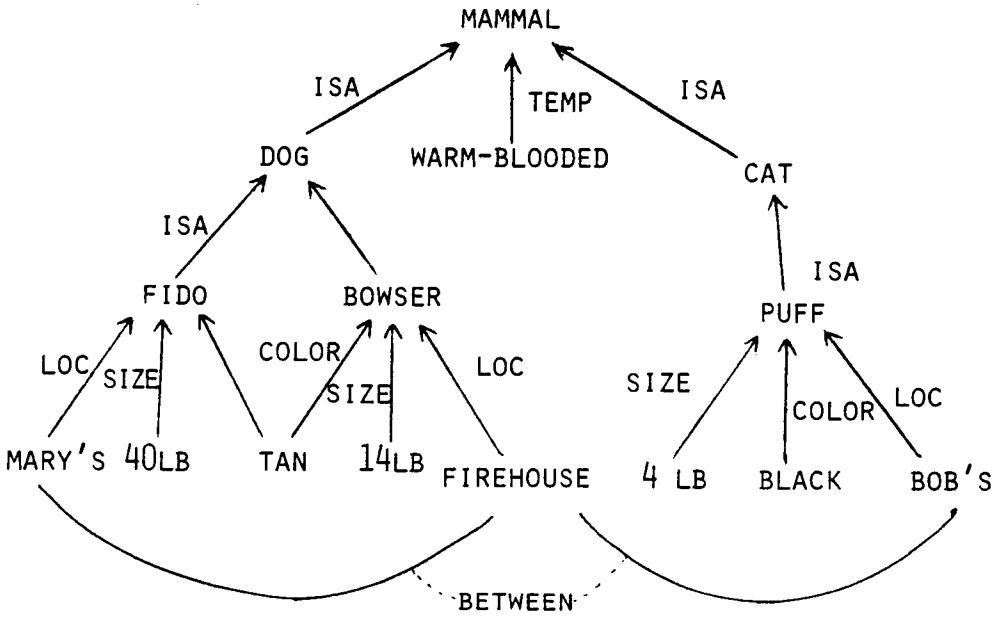
A semantic network is a KS. It is built up from knowledge chunks that are instances of a relation. The format of a chunk is: rel (a1 ... an), where rel is a relation name and the ordered tuple, (a1 ... an), is in the relation rel. For example, ISA (Dog, Mammal) means (Dog, Mammal) is a member of the relation ISA. ISA is conventionally taken to be the relation, more-specific-exmaple-of. Thus, the above is the representation of the fact that a Dog is a specific kind of Mammal.

| NASA |

RELATIONS

TEMP(WARM-BLOODED MAMMAL)
ISA(DOG,MAMMAL) ISA(CAT,MAMMAL)
ISA(FIDO,DOG) ISA(BOWSER,DOG) ISA(PUFF,CAT)
LOC(MARY'S,FIDO) LOC(FIREHOUSE,BOWSER) LOC(BOB'S,PUFF)
COLOR(TAN,FIDO) COLOR(TAN,BOWSER) COLOR(BLACK,PUFF)
SIZE(40LB,FIDO) SIZE(14LB,BOWSER) SIZE(4LB,PUFF)
BETWEEN(MARY'S,FIREHOUSE,BOB'S)

SEMANTIC NETWORK



RULES OF INFERENCE

ISA(X,Y) ^ ISA(Y,Z) => ISA(X,Z)

SIZE(X,Y) ^ SIZE(U,V) ^ X<U => SMALLER(Y,V)

ISA(X,Y) ^ R(U,Y) => R(U,X)

FIGURE 3.8 EXAMPLE SEMANTIC NETWORK

Figure 3-8 shows a semantic network (or net). The top of the figure lists the instances of relations using the relation names TEMP, LOC, COLOR, SIZE, ISA, and BETWEEN. TEMP(a,b) means a is the temperature of b; LOC(a,b) means a is located at b; COLOR(a,b) means that a is the color of b; and SIZE(a,b) means a is the size of b. The knowledge in a semantic net is given meaning by defining the relation names and other symbols used in the instances of relations, in terms of external entities.

The graph in the middle of Figure 3-8 shows exactly the same knowledge that is in the set of instances at the top of the figure. The entity names are connected by arrows labeled with appropriate relation names.

The external format of knowledge in a semantic network is usually very similar to the one used here. However, the internal storage of the semantic network closely corresponds to the graphical presentation - that is, a network structure built using pointers and list structures. The explicit connections among the entities enhances efficiency of programs that search through the semantic network.

The bottom of Figure 3-8 gives some examples of inference rules for the semantic network. It is possible to represent the inference rules as a production system. This has the advantage

of allowing procedural knowledge to be used to test for complex conditions.

The first rule says that (for all x, y, and z) if x is a y and y is a z, then x is also a z. An example of this is: Fido is a Dog and a Dog is a Mammal; therefore, Fido is a Mammal. The second rule says that if y and v are two entities that have SIZE, and the size of y is less than the size of v, then y is SMALLER than v. Thus the instance of the relation, SMALLER(Puff, Bowser).

This inference rule defines instances of relations whose names do not appear explicitly in the semantic network. The last example inference rule says that, if x is a y, and y has a property conferred by the binary relation, r, then x has the same property conferred by r, i.e., properties are inherited. Thus, Fido is a Mammal (by the transitivity of ISA - first rule), and a Mammal has the property, WARM-BLOODED (conferred by the relation TEMP), therefore, Fido is WARM-BLOODED.

However, the indiscriminate use of the third rule can cause derivation of incorrect results. For example,

ISA(Dog, Mammal) AND ISA(Cat, Mammal) ==> ISA(Dog, Cat)

In order to avoid this kind of problem, it is necessary to have some knowledge (non-syntactic) about the relations to which inference rules are applied. One possible solution is to have a rule, such as the third example rule, for each relation that is inheritable. Another solution is to embed the inference rules in the CE along with the necessary ad hoc knowledge to avoid the problems. If the number of relations occurring in the semantic network is large or if the relation set can be modified or expanded - then both these approaches cause problems.

A more general approach to the problem treats relation names and entity names more uniformly. For example, temperature is defined as an inheritable property by an instance like INHERITABLE(TEMP).

The third inference rule is then rewritten as:

ISA(x,y) AND r(u,y) AND INHERITABLE(r) ==> r(u,x)

With this approach, relations can be arguments to relations, and hence have the same properties as other entities.

Another choice and tradeoff about a semantic network is the decision about which relations and which instances in the relations ought to be stored explicitly and which should be

computed via the inference rules. Explicit storage costs space, and inference rules cost computation time.

A technique often used with semantic networks is to make a distinction between general knowledge and specific knowledge and to store the two in a different manner. Specific knowledge has the general characteristic of being "low" in the tree - as shown in the middle of the figure. This means:

(1) There are few if any chains below it.

(2) Therefore, properties have simple values.

(3) Most entities in the same general classification have all and only a known set of properties.

(4) There are a large number of entites in a general class. For our example, the specific knowledge can be displayed tabularly as:

ENTITY	ISA	SIZE	COLOR	LOC
Fido	Dog	401lb	Tan	Mary's
Bowser	Dog	141lb	Tan	Firehouse
Puff	Cat	41lb	Black	Bob's

The above conditions make it likely that the specific knowledge can be gathered into a tabular form by simple

mechanical means, and that the specific knowledge (which is usually most of the semantic net) can be kept in relatively inexpensive secondary storage and even accessed through an efficient, existing data management system. The general knowledge is kept in primary memory. Fortunately, most processing by the inference rules occurs on other than the "bottom" of the network, so that efficiency is maintained.

Semantic networks are by far the best available technology for representing definitional and relational knowledge that is too complex for ordinary data management techniques. This is the case because the structure allows for the inclusion of ad hoc information, and the utilization of inference rules permits straightforward enhancement of the inherent representational power and completeness.

On the other hand, there are some disadvantages to the use of semantic networks to represent knowledge in a KBS. The principal one is that the chunk size is fairly small. The result of this is that reasoning chains can be quite lengthy and tedious. Another result is that processing a semantic net can consume large amount of computer time.

Additional references include the following: [Duda, et.al, 77; Grignetti, 75; Mylopaulos, et.al, 75; Norman, 75; Woods, 75;

Schubert, -76].

3.1.6 Frames

A research topic of great current interest in computer representation of knowledge is frame theory. No one has succeded in defining frames to all researchers' satisfaction. Some of the common features in proposals about frames are:

(1) A frame is a knowledge chunk.

(2) A frame has a collection of definitional and procedural knowledge about an object, action, or concept.

(3) A frame is a complex data structure that has named slots corresponding to definitional characteristics.

(4) A frame has the ability to attach procedural knowledge to the slots and/or to the frame itself.

An example (Figure 3-9) to illustrate some of the above features is discussed below.

The top of the figure presents definitional information about a dog. The first line states that a dog is a "mammal". The next line means that there is a slot, named "kind" (of dog), that may be filled with a value of (type) "breed". ("Breed" in this example is itself a frame). The color of a dog is limited to one or a mixture of the stated colors by the SUBSET.OF

operator. Default values are indicated by underlining, and the FROM operator is used to pick out values from other frames. Thus, the combined effect of the phrase FROM color OF kind is to make the default value for the color of a dog the default for his breed. The state of a dog is either "adult", the default, or "puppy" if the age is known to be less than one year.

The bottom of the figure shows a frame for "boxer" and declares that boxer is a breed - but only a breed of dog. The color of a boxer is restricted to one of the colors "tan", "brown", and "brindle", with a default of "tan". If this breed did not have a characteristic color restriction, then this slot would be omitted. The next slot says that the size of a boxer is between 40 and 60 pounds. No default is specified. The tail and ears slots are defined with default value "bobbed" and the respective alternatives of "long" and "floppy".

```
dog       FRAME   ISA   mammal
          kind    breed
          color   SUBSET.OF {tan brown black white rust}

          FROM color OF kind

          leggedness      0...4
          weight          >0,  FROM size OF kind
          state           adult  OR  puppy if age < 1
          age             >0,  now birthday
          birthday        date
          name            string

          END             dog


boxer     FRAME   ISA  breed OF dog
          color          ONE.OF {tan brown brindle}
          size           40...60
          tail           bobbed OR long
          ears           bobbed OR floppy
          temperment     playful
          COMPLAINTS     IF weight > 100 THEN ASSUME
                                              (great dane)
          END            boxer
```

Figure 3-9 EXAMPLE FRAME DEFINITIONS

-Temperment is shown to always be playful. The last line shows an example of a complaint and ad hoc knowledge used to make a recommendation, namely, if you see a giant boxer, then assume that it might be a Great Dane instead.

Figure 3-10 shows an example use of frames in a recognition task. The top of the figure shows some feature values (e.g., color is tan, ears are bobbed) that have been detected for an object, here identified as number 456. The CE has matched the known feature values with the available frames and has manufactured the working hypothesis shown at the bottom of the figure - namely, a boxer dog that is assumed to have bobbed tail and to be an adult. It is noted that this particular boxer (object number 456) is mean and that this is exceptional. Also, the size of the boxer was only approximately known, but the approximation has been used in lieu of a more accurate value.

LOW-LEVEL INFORMATION

OBJECT 456

```
color       = tan
ears        = bobbed
leggedness  = 4
size        = 40 - 45
temperment  = mean
```

TRIAL IDENTIFICATION

```
[OBJECT  456 ISA dog

    kind         boxer   WITH [color        tan
                               size         40 - 45
                               tail         ASSUMED bobbed
                               ears         bobbed
                               temperment   EXCEPTIONAL
                                            mean]
    color        tan
    leggedness   4
    weight       40 - 45
    state        ASSUMED adult]
```

Figure 3-10 INEXACT MATCH BY A FRAME SYSTEM

A possible scenario for the recognition task: A general matching procedure would attempt to instantiate all frames in the system until a reasonable fit was found; in the example, "boxer" is a reasonable match. Slots that are yet unfilled would be used to hypothesize other values not yet detected. For the boxer frame, a bobbed tail would be predicted and put on an agenda of things to look for. Assuming there was a frame for tails, it might possibly contain heuristic knowledge about how to more carefully scan the raw data to confirm or deny the existence of a particular kind of tail. Other activity that could emanate from the boxer frame is the activation of a complaint. Thus, if the weight of the boxer was too large, the complaint mechanism could change the identification of the instantiation of the boxer frame into one for a Great Dane.

Besides the prediction and correction activity resulting from the partial match to a frame, a third process can be tried. Namely if the match is good enough, then the frame can become more informative. For our example, the transformaton is from a boxer to a boxer dog where more information is absorbed, e.g., leggedness.

The belief is that this style of recognition will be more goal directed - and hence more accurate and efficient -

than general techniques that depend upon regularity and uniformity of structure.

Additional relevant references include the following: [Nilsson, 80; Winston, 77; Bobrow, 77; Winograd, 75; Malhotra, 75; Davis, 76].

3.2 Workspace Representation

The workspace is the encapsulation of the system's current state in a problem solving activity. It includes:

(1) Global variables - computed values, goals, and input problem parameters.

(2) An agenda - a list of activities that can be done next.

(3) A history - what has been done (and why) to bring the system to its current state.

The simplest example of a workspace representation is the push-down stack in a LISP-like system. The stack contains the bindings of global variables, return address (a history snapshot), and the values of temporaries. There is no agenda in a simple system other than the program counter. A more

complicated example is the data base in a production system. It contains the entire state of the system, including an implicit agenda (the conflict set of rules that can apply).

In most computer programs, the workspace can be represented in an ad hoc way using whatever techniques are provided by the containing proram-language system. However, this is not always adequate in KB system because:

(1) The capability to provide explanations is based in part on an ability to find the trace of events (history) that produced the solution.

(2) Efficient reasoning behavior depends on the selection of the next thing to be done - hence, the necessity for an explicit agenda and visibility of enough state (global variables) to make informed decisions. Further, if a KBS has many knowledge sources, the workspace representation may be used to provide a communication channel among them.

3.3 Cognitive Engine

In a KBS, the primary function of the cognitive engine (CE) is to perform the task of problem solving. A secondary function

of the CE is to explain the behavior of the system and support its derived solutions. To accomplish these functions, the CE must:

(1) Control and coordinate system activities and resources.

(2) Plausibly reason about domain-specific problems by having access to and using the contents of the KB, the contents of the workspace, and knowledge and procedures embedded in the CE.

(3) Link the KB with the interface module(s).

The CE is the most active component of a KBS. That is, it is always instantiated, and it controls the activation and utilization of other system modules. Another characterization of a CE is that it is the intelligence or understanding component of a KBS (even though its activity may, to a degree, be guided by higher-level (control and/or meta) knowledge in a KS).

The following three definitions are from [Feigenbaum, 81].

A CE is <u>sound</u> if it produces only correct or "I don't know" solutions, i.e., it does not produce incorrect solutions.

A CE is complete if it can always produce a solution to a posed problem when a solution exists.

A CE is admissible if it always finds a minimal-cost solution when a solution exists. The cost is taken to mean the cost of using the solution, not necessarily the cost of finding it.

3.3.1 CE Strategies

The input to a CE is usually a set of initial conditions and a goal. The KB is used in some manner to find a method of obtaining the goal given the constraints imposed by the initial conditions. There are four ways of doing this:

(1) Forward chaining - apply the KB to the givens to infer new conditions; continue in this manner until the goal is satisfied.

(2) Back chaining - apply the KB to the goal to produce new subgoals; continue in this manner until the initial constraints or primitive conditions (known to be solvable) are reached.

(3) Chain both ways - forward chain from the initial conditions and backward chain from the goal until a

- common middle term is produced.

(4) Middle term chaining - using the KB, guess a middle term and solve separately the problem of getting from the initial conditions to the middle term and from the middle term to the original goal; continue in this manner until a solution in terms of primitives is generated. (This method is also called problem reduction.)

Figure 3-11 shows an example of the first three of these techniques. The KB contains three rules:

(1) Any integer, x, can be replaced by 2x (x --› 2x).

(2) Any even integer, 2x, can be replaced by x (2x --› x).

(3) Any integer, x can be replaced by 3x+1 (x --› 3x + 1).

The problem is to transfer 4 into 20 using the permitted operations. The top figure shows forward chainging - i.e., start with 4 and apply the operations until 20 is produced. The middle figure shows back chaining - i.e., start with the goal, 20, and use the inverse of the above rules and continue until 4 is produced. The bottom figure shows the chain-both-ways technique.

First, one step of back chaining produces the nodes labeled 10 and 40. Then one step of forward chaining produces the nodes labeled 8,2, and 13. Finally, one more step of back chaining is done to produce the nodes labeled 5,3,13, and 80. Since 13 is on both the forward and backward grown "wave fronts", the process can terminate; otherwise, the steps of forward and backward chaining would continue to alternate until either a solution was found or the system gave up.

| N A S A |

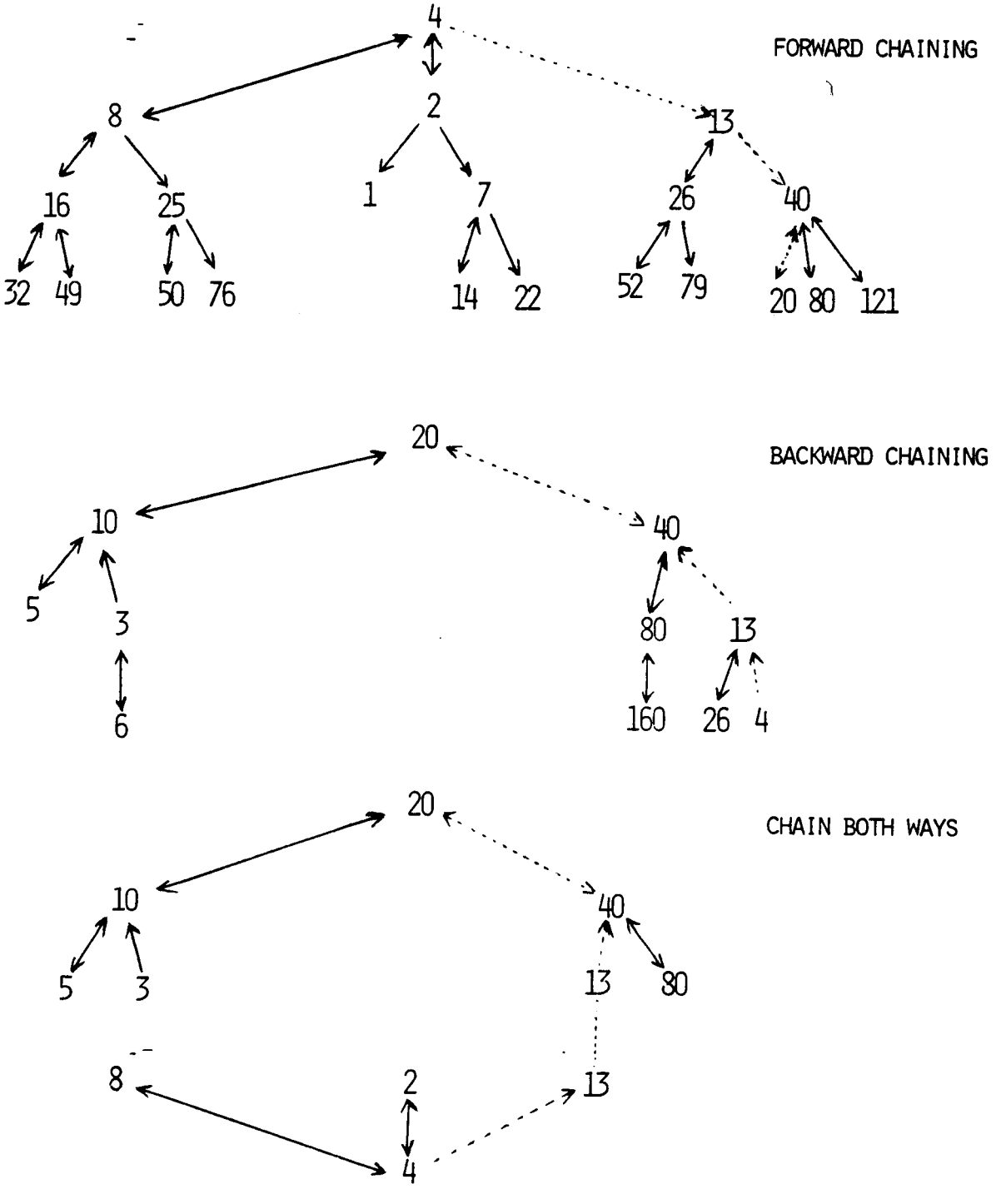


FIGURE 3.11 CHAINING EXAMPLES

Another way of differentiating CE strategies is via breadth-first vs. depth-first . In a breadth-first system, all possible methods of continuing are attempted in parallel. This is showed in Figure 3-11, where each (horizontal) level of the graph was generated by a single cycle of the system. In a depth-first system, some path (node, state, etc.) is selected and a single continuaton is attempted, i.e., the node is not fully expanded all at once. This path continues growing until either the path reaches a solution or some path-length constraint is violated. In the latter case, the path is backed up to the deepest node at which an alternative exansion exists. At that point, another path continuation is generated. This process continues until either a solution is produced or the alternatives that could produce a solution within the length constraint are exhausted. A depth-first strategy can be more efficient than a breadth-first one if a good technique exists for ordering production of path extensions. Figure 3-12 shows an example of a depth-first strategy combined with back-chaining for the prior problem. A maximum path length 4 was used as a constraint, and the order of (inverse) operator application was:

(1) n --› 2n.

(2) 2n --› n.

(3) - n --› 3n + 1.

Each node has a superscript that denotes the order in which the nodes were generated.

Additional references include the following: [Nilsson, 80; Klahr, 78; Miller, 73].

| N A S A |

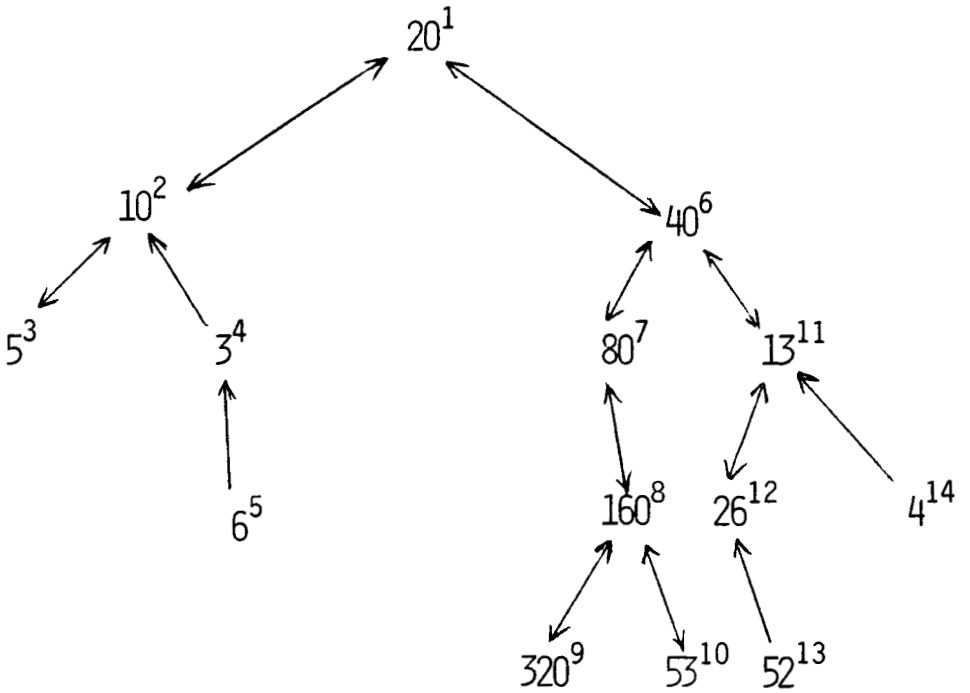


FIGURE 3.12 DEPTH-FIRST BACK CHAINING

3.3.2 Methods of Implementing the CE

Most methods and techniques used to implement a CE are intimately connected to the choice of a representation technique for the KB. However, there are a few methods that are general enough to be used with a variety of KB representations. Two classes of them are: search methods and modeling/simulaton methods [Feigenbaum, 81]. The third widely used technique is pattern-matching [Kanal, 68].

These techniques will be addressed in a future report.

3.4 The Interface

The interface component of a KBS provides the necessary connections and communications with the environment and user. It is not always engineered as a separate module in the system; usually it is integrated into the CE and accesses the KB [Davis, 76].

The interface has three logical parts: the user interface, the expert interface, and the external data interface.

3.4.1 The User Interface

The user interface is the most important component of a KBS in determining its acceptability to the practitioners of the intended domain. In general, they are neither computer scientists nor programmers, and may view the computer, especially a KBS, as a feared and unworthy competitor. A properly functioning user interface will smooth out and minimize the problems associated with learning any new system, and in the long run improve system productivity by making it possible for the users to be more cooperative in problem-solving activities.

In order to qualify as a KBS, the user interface must, at a minimum, be interactive and use domain-specific jargon. The reason for requiring the system to be interactive is simply that the state of the art does not provide techniques for going from problem statement to "best" answer without additional information that must be solicited from the user. The problem with providing initially, along with the problem statement, all information that might conceivably be needed, is that most of it is unnecessary. Another reason for wanting a KBS to be interactive is so that explanations of system behavior and results can be solicited.

Besides using domain-specific jargon, many KB systems accept and output information using an English-like natural language.

Much of the ambiguity of natural language can be eliminated when the dialog is restricted to a particular domain and it is known that the user is engaged in goal-oriented problem-solving activity.

Another desirable characteristic of the user interface is soft failure. That is a KBS should not blow up because the user makes a mistake, nor should it conceal its problem. A useful technique to smooth over some problems of this sort is a spelling corrector.

One of the problems confronting the explanation mechanism in the user interface is translating explanations into a natural form for the user. Fortunately, the complexity of this task is substantially reduced by uniformities of format in the KB and in the workspace. For example, in a production system, it is straightforward to turn an IF-THEN rule out in reasonable English. The usual approach is to have a separate scheme for each kind of knowledge chunk in the KB and element in the workspace; most such schemes look like sophisticated fill-in-the-blank formulae.

3.4.2 The Expert Interface and Knowledge Acquisition

The expert interface is the mechanism through which

knowledge is added to the KB or the KB is modified. Its intended users are experts in the problem domain and the system implementors who are responsible for building the initial KB. This interface is often called a knowledge acquisition interface [Barnett, 75]. Unlike the user interface, it can be assumed that the user of the expert interface has some knowledge and awareness of the structure and functioning of the KBS. Table 3-1 summarizes the knowledge-acquisition process.

The knowledge that goes into a KBS must originate from some external source. The most usual is an expert in the problem domain. He can provide specific facts, rules of thumb, and the rules of reasoning, along with his rating of confidence. Other originating sources of knowledge commonly used are journal articles, texts and engineering handbooks. The information from these sources often is hard data and tabular. Therefore, it usually comprises fact files in the KB.

ORIGINATING SOURCE

- Domain Expert (or User)
- Journals
- Texts
- Protocol Studies
- Derived Results

ENTERED BY

- Domain Expert
- Implementor
- User
- CE

COMPILED BY

- Knowledge Acquisition Interface
- Implementor

ISSUES

- Interface Language
- Consistency
- Accommodation
- Confidence Factors

MAJOR OUTSTANDING PROBLEMS

- Knowledge Acquisition
- Learning
- Extensibility

TABLE 3-1 KNOWLEDGE ACQUISITION
(Based on [Hayes-Roth, 83])

Ideally, the knowledge-acquisition interface can be used by domain experts and users of the system other than the implementation staff. However, in many KB systems, the complexities of adding to or modifying the KB are such that programming skills are required. For such systems, a computer specialist may need to act as an intermediary between the originating source and the KBS.

The knowledge-acquisition interface has three major tasks:

(1) Accepting knowledge in external format and translating it into internal format.

(2) Validating consistency of new and old knowledge.

(3) Storing knowledge into the KB. This is called Compilation.

4. APPLICATION CONSIDERATIONS [Buchanan, 75; Nilsson, 80]

Though there exists a relatively diverse collection of existing and developing KBS applications, the selection process for each new application requires consideration of a variety of issues. First, there is a set of initial considerations that address the issues of the problem domain itself and the people

associated with it, the experts and the practitioners. Next, are the technology considerations that focus on the availability of usable technology for implementing a KBS that has successfully met the first criterion. Last, there are the equally important considerations that are directed at determining whether or not the development environment and the user environment are properly supportive.

Each of these groups is considered below in the form of a set of questions [Buchanan, 75].

4.1 Initial Considerations

(1) Does the problem have a closed-form solution?

(2) Is there an expert who knows how to solve problem?

(3) Is there an expert available and can he work on the development of a KBS?

(4) Does the expert have a model to solve the problem?

(5) Is the domain well bounded?

(6) Are the intended users professionals?

(7) Are the economics right?

4.2 Technology Considerations

(1) Can a first-order model (simulation) be constructed with the help of expert(s)?

(2) Is there a knowledge representation that matches the "chunk size" of the expert's knowledge?

(3) What are the necessary knowledge source(s) and their representation(s)?

(4) What reasoning or inference methods are needed?

(5) Are the knowledge representation and reasoning method compatible with one another?

(6) Will the system support growth?

4.3 Environmental Considerations

(1) Is there an interactive system for the KBS users?

(2) Do the necessary tools exist, in particular a properly expressive programming system?

(3) Will the resultant system perform effeclently?

5. CONCLUSIONS

The technology of knowledge-based systems has emerged from the laboratory, but it has not achieved the status of being a commonly known way of implementing computer-based application systems. Systems have been developed in an intriguing spectrum of application areas, from medicine and chemistry to geology and business. The general level of accomplishment appears to be high enough to make it worthwhile to begin exploring other areas for immediate potential application.

There remain a number of unresolved issues. Even with a group of domain experts who are cooperative and well motivated, the methodology for transferring their knowledge to the system is, at best, ad hoc. This is the area in which more research is needed to discover (or convert) what amounts to a completely new technology: the acquisition, communication, and representation of expertise (the ability to use a body of knowledge effectively in solving a particular problem).

APPENDIX A. A KBS CASE STUDY (MYCIN)

MYCIN is a medical consultation system. The material covered here is a condensation of [Shortliffe 76; Shortliffe 75; Davis, 77; Davis, 76].

The Problem Domain And The Users

MYCIN is a knowledge-based interactive computer system intended to provide advice to physicians on prescribing antimicrobial therapy for bacterial infections of the blood (bacteremia). The problem of therapy selection and recommendation for an infectious disease is difficult and complex. The first is to determine whether or not the infection is serious enough to warrant treatment. If it is determined that treatment is warranted, one should know what organism is causing the infectious disease. To do this, one must obtain a specimen of the infection for culturing, analysis, and identification by a laboratory. This is a time consuming process. And in many cases, the infection is serious enough that treatment must be begun before all of the analysis can be completed. Therefore, any recommended therapy must be based on incomplete information. To further complicate matters, the most effective drug against

the suspected or identified organism may be totally inappropriate for the specific patient because of age or medical conditions and problems. Thus, any system or consulting physician must be aware of all of these complexties if proper advice is to be rendered in each specific case. MYCIN has been designed to cope with just such complexities and interrelationships among the many variables and to provide a physician with advice that is proper for each individual patient.

Though the problem is quite complex, the domain is well bounded. MYCIN requires knowledge related only to infectious diseases, and knowledge related to experience with various infectious organisms in terms of resistance to specific drugs, and knowledge of symptoms related to specific infections.

MYCIN is intended to be used by physicians. The dialogue that it carries on with the user is in the jargon of medicine and specifically that of infectious diseases, laboratory procedures, infectious organisms, drugs, etc. Thus, a user is expected to be a competent medical practitioner.

MYCIN'S Knowledge Base

MYCIN's knowledge base (KB) contains several knowledge sources - production rules, clinical parameters, special functions, and procedures for therapy selection.

Each production rule consists of a Premise, which may be a condition or a conjunction of conditions, an Action to be taken, and sometimes an Else clause. For the action to be taken, each of the conditions in the Premise must hold. If the truth of the Premise cannot be ascertained or the Premise is false, the action in the Else clause is taken if one exists; otherwise, the rule is ignored. In addition, the strength of each rule's inference is specified by certainity factor (CF) in the range -1 to +1. CF's will be discussed in the next section. Each rule in MYCIN falls into one and only one of the following categories:

(1) Rules that may be applied to any culture.

(2) Rules that may only be applied to current cultures.

(3) Rules that may be applied to current organisms.

(4) Rules that may be applied to any antimicrobial agent administered to combat a specific organism.

(5) Rules that may be applied to operative procedures.

(6) Rules that are used to order the list of possible therapeutic recommendations.

(7) Rules that may be applied to any organism.

(8) Rules that may be applied to the patient.

(9) Rules that may be applied to drugs given to combat prior organisms.

(10) Rules that may be applied only to prior cultures.

(11) Rules that may be applied only to organisms isolated in prior cultures.

(12) Rules that store information regarding drugs of choice.

The system also contains a collection of clinical parameters, represented as ‹attribute, object, value› triples. These clinical parameters fall into six categories:

(1) Attributes of cultures.

(2) Attributes of administered drugs.

(3) Attributes of operative procedures.

(4) Attributes of organisms.

(5) Attributes of the patient.

(6) Attributes of therapies being considered for

recommendation.

Each of the parameters has a certainty factor reflecting the system's "belief" that the value is correct and an associated set of properties that is used during consideration of the parameter for a given context. These properties specify such things as the range of expected values a property may have, the sentence to transmit to the user when requesting data from him, the list of rules whose Premise references the parameter, the list of rules whose Action or Else clause permits a conclusion to be made regarding the parameter, etc. Only those properties that are relevent to each parameter are assocated with it. However, properly specifying how the parameter is to be represented in English is mandatory for all.

Additional information is contained in simple lists that simplify references to variables and optimize knowledge storage by avoiding unnecessary duplication. These lists contain such things as the names of organisms known to the system and the names of normally sterile and non-sterile sites from which organisms are isolated.

In conjunction with a set of four special functions, MYCIN uses knowledge tables to permit a single rule to accomplish a task that would otherwise require several rules. The knowledge

tables contain a record of certain clinical parameters and the values they may take on under various circumstances.

There is one knowledge source in MYCIN that is as a set of functions. This is the knowledge required for choosing the apparent first-choice drug to be recommended.

This constitutes the majority of MYCIN's knowledge base, which permits the system to comprehend the nature of an infection without complete information about the organism involved and provide the physician with proper advice regarding treatment under the circumstances. This organization and structure, along with the way the knowledge is used facilitates the system's ability to explain its actions and advice.

MYCIN's Cognitive Engine

MYCIN's cognitive engine is domain independent in the sense that none of the knowledge required to proide advice about bacteremia is embedded in it. Thus, additional rules concerning infectious disease may readily be added, or a new knowledge base could be substituted to provide therapeutic advice about a different domain of infections.

The task that MYCIN performs, under the control of its CE, can be characterized as a four-step decision process:

(1) Decide which organisms, if any, are causing significant disease.

(2) Determine the likely identity of the significant organisms.

(3) Decide which drugs are potentially useful.

(4) Select the best drug or drugs.

A consultation session between a physician and MYCIN results from a simple two step procedure:

(1) Create the patient "context" as the top node in the context tree.

(2) Attempt to apply the "goal-rule" to the newly created context.

The "goal-rule" is one of the rules from the category of those that may be applied to the patient (as described above), and states: "If there is an organism that requires therapy and consideration has been given to the possible existence of additional organisms requiring therapy, even though they have not been recovered from any current cultures, then do the following: Compile a list of possible therapies which, based upon

sensitivity data, may be effective against the organisms requiring treatment and determine the best therapy recommended from the compiled list; otherwise, indicate that the patient does not require therapy."

This rule follows the four-stage decision process given above.

The two components or programs that constitute MYCIN's CE are called MONITOR and FINDOUT. MONITOR's function is to determine whether the conditions stated in the Premise of a rule are true. To do so, it considers each condition of the Premise at hand, first determing whether it has all of the necessary information to make the determination. If it requires information, it calls FINDOUT to obtain what is needed. FINDOUT first determines whether the needed information is laboratory data. If it is, it asks the physician for it. If the physician cannot provide it, FINDOUT retrieves the list of rules that may aid in deducing the information and calls MONITOR to evaluate the rules. When the process completes, control is returned to MONITOR. If the information needed is not laboratory data, FINDOUT retrieves the list of rules that may aid in deducing the needed information and calls MONITOR to evaluate the rules. If the deductive process of applying the rules (backward from a goal to the data or information needed) cannot provide the needed

information, the physician is asked to provide it. In either case, control is returned to MONITOR. Given the information that is provided by FINDOUT or that was already available, MONITOR determines whether the entire Premise is true. If it is not, and there is no Else clause, the rule is rejected. If the Premise is true or the Else clause is invoked, the conclusion stated in the Action of the rule or in the Else clause is added to the ongoing record of the consultation, and the process completes. Note that there is a recursive relationship between MONITOR and FINDOUT, such that so long as any information is needed to evaluate a Premise, or rules are required to develop the needed information, the two components are in a recursively dependent and oscillating relationship until the very first rule invoked, the "goal-rule", is satisfied. In the process of evaluating the rules, a great deal of related and necessary information and data are developed and retained in various tables and structures in the workspace. They serve two purposes:

(1) They prevent wasted effort that would be required to redevelop information that has already been obtained, and to prevent the system from endlessly chasing its tail.

(2) They provide the necessary history required for the

explanations that may be requested by the user.

In addition to having certainty factors (CFs) for the rules and the clinical parameters in the knowledge base, the physician, when asked for either laboratory data or other information that the system itself cannot deduce, may attach a CF to his input. The default, if the physician does not provide a CF, is assumed to be +1. The certainty factors are the key to permitting MYCIN to perform inexact reasoning. The rationale, mathematics, and application are thoroughly treated in [Shortliffe, 76]. The presentation here is very simplified.

A certainty factor (CF) is a number between -1 and +1 that reflects the degree of belief in a hypothesis. Positive CFs indicate that there is evidence that the hypothesis is valid; the larger the CF, the greater the degree of belief. A CF = 1 indicates that the hypothesis is known to be correct. A negative CF indicates that the hypothesis is invalid; CF = -1 means that the hypothesis has been effectively disproven. A CF = 0 means either that there is no evidence regarding the hypothesis or that the evidence is equally balanced. The hypotheses in the system are statements regarding values of clinical parameters for the nodes in the context tree. To properly perform, MYCIN must deal with competing hypotheses regarding the value of its clinical

parameters. To do so, it stores the list of competing values and their CFs for each node in the context tree. Positive and negative CFs are accumulated separately as measures of belief (MB) and measures of disbelief (MD) and added to form a resultant CF for a clinical parameter. The CF of a conclusion is the product of the CF of the rule that generated the conclusion and the tally of the CFs of the clinical parameters that were used in substantiating the conclusion. When a second rule supports the same conclusion, the CFs are combined by $z = x + y(1-x)$, where x is the CF of the first supporting rule, y is the CF of the succeeding rule and z is the resultant CF for the conclusion. The CFs permit the system to report findings to the physician with varying degrees of certainty such as, "There is strongly suggestive evidence that ....", "There is suggestive evidence that ....", "There is weakly suggestive evidence that ....", etc.

Context Tree. The topmost tree is always the patient. Branches are added successively to the existing nodes as FINDOUT discovers a need for them in attempting to obtain requested information for MONITOR. Thus, given only the patient, when MONITOR requests information from FINDOUT about organisms in order to evaluate the first condition in the Premise of the goal-rule, FINDOUT discovers that it cannot get organism information without having information about cultures. Thus,

context(s) concerning cultures(s) are spawned from the patient node, from which eventually are spawned contexts for the organisms identified by the cultures. For those organisms deemed significant, links attach to context nodes about the relevant drugs for treating these organisms. Thus, the context tree is tailored for each patient as the system progresses through its reasoning process.

MYCIN's Explanations

One of the primary design consideration taken in MYCIN was the requirement that the system be able to explain its decisions if physicians were going to accept it. Selecting rules as the representation of the system's knowledge greatly facilitated the implementation of this capability. The physician using the system enters the explanation subsystem automatically when the consultation phase is completed, or he may enter it upon demand during the consultation session at any point at which the system requests input from him. In the latter case, he can input "WHY" to request a detailed answer about the question just asked of him or he can input "QA" to enter the general question-answering explanation subsystem to explore the decisions and other aspects of the consultaion up to the point of divergence.

The explanation provides several options to the physician.

Since the system automatically enters this mode at the end of the consultation, the physician may simply input "STOP", which terminates the system. He may input "HELP", which provides him with the list of explanation options, which include:

Input	Option
EQ	Explain a specific question asked of the physician during the consultation - each has a sequence number, which must accompany the EQ request.
PR	Requests a particular rule be printed and must be followed by the rule number.
IQ	Is a prefix for a question about information acquired by the system during the consultation. The question is phrased in the limited English that MYCIN can handle.
no prefix	A general question is assumed being asked about the content of MYCIN's rules.

Thus, the physician can ask to have Question 48 explained by inputting "EQ48". To which the system would respond: QUESTION 48 WAS ASKED IN ORDER TO FIND OUT THE PATIENT'S DEGREE OF SICKNESS (ON A SCALE OF 4) IN AN EFFORT TO EXECUTE RULE068. He may then optionally input "PR68" or "WHAT IS RULE068" to see what exactly was being sought and why.

MYCIN's Interfaces

MYCIN has two interfaces. One is for the using physician,

through which he may answer questions posed by the system and ask questions of it; the other is a knowledge-acquisition interface accessible only to experts recognized as such by the system.

All of the questions asked of the user have been carefully designed not to require the language-understanding component. Thus, instead of asking, "What is the infectious disease diagnosis for the patient?" it asks, "Is there evidence that the patient has a meningitis?" To which only a simple "yes" or "no" is required.

The knowledge-acquisition interface, on the other hand, permits the expert to input a new rule in stylized English, with prompting to obtain the rule in the proper sequence: Premise first, condition by condition, followed by the Action, and then an Else clause if one is required. The system then translates the rule into internal form, reordering the conditions of the Premise if necessary, according to a set of criteria developed to improve the rule-evaluation process. It then retranslates the rule into English and requests that the expert decide whether the rewritten--version was the one intendend. If not, the expert may modify selected parts and is not required to restate the entire rule unless there has been a gross misunderstanding.

The same mechanism is used when an expert wants to correct

or modify an existing rule. In all cases, when a new or corrected rule has been approved by the expert, the system checks to see whether the rule is consistent with the existing rule set. If the new or modified rule subsumes or is subsumed by an existing rule, it is not readily discoverable, and no test is made for this condition. If a rule is discovered to be in conflict with an existing rule, it is rejected.

REFERENCES

[Bobrow, 77]. D. Bobrow, T. Winograd, "An Overview of KRL, a Knowledge Representation Language", Cognitive Science, vol. 1, no. 1, 1977.

[Barnett, 75]. J. Barnett, "A Phonological Rules System", Technical Memo, TM-5478/000/00, System Development Corporation, Santa Monica, CA, 1975.

[Brachman, et.al, 80]. R. Brachman, B. Smith, "Special Issue on Knowledge Representation", Sigart Newsletter, no. 90, Feb. 1980.

[Buchanan, 75]. B. Buchanan, "Applications of Artificial Intelligence to Scientific Reasoning", Proc. Second USAJapan Computer Conference, Tokyo, Japan, 1975.

[Chandrasekaran, 79]. B. Chandrasekaran, "An Approach to Medical Diagnosis Based on Conceptual Structures", Proc. Sixth Int'l. Joint Conf. Artificial Intelligence , 1979.

[Chilausky, 76]. R. Chilausky, B. Jacobsen, R. Michalski, "An Application of Variable-Valued Logic to Inductive Learning of Plant Disease Diagnostic Rules", Proc. Sixth Annual Int'l. Symp. Multiple-Valued Logic, 1976.

[Davis, 76]. R. Davis, "Application of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases", Stanford AI Laboratory Memo AIM-283, Report no. STAN-CS-76-552, Stanford University, Stanford, CA, 1976.

[Davis, et.al, 77]. R. Davis, B. Buchanan, and E. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program", Artificial Intelligence, vol. 8, no. 1, 1977.

[Davis, 81]. R. Davis, "The Dipmeter Advisor: Interpretation of Geological Signals", Proc. Seventh Int'l. Joint Conf. Artificial Intelligence, Aug. 1981.

[Feigenbaum, 81]. E. F. Feigenbaum, A. Barr, The Handbook of Artificial Intelligence, Vols. 1,2,3, Harris Tech Press, Stanford, Calif., 1981.

[Feigenbaum, 71]. E. Feigenbaum, G. Buchanan, J. Lederberg, "Generality and Problem Solving : A Case Study Using the DENDRAL Program", Machine Intelligence vol. 6, D. Meltzer and D. Michie, (Eds.), Edinburgh Univ. Press, Edinburgh, Scotland, 1971.

[Forgy, 76]. C. Forgy, "A Production System Monitor for Parallel Computers", Computer Science Dept. Technical Report, Carnegie-Mellon Univ., Pittsburgh, PA, 1976.

[Forgy, 77]. C. Forgy, J. McDermott, "OPS, a Domain-Independent Production System Language", Proc. Fifth Int'l. Joint Conf. Artificial Intelligence, pp 933-939, 1977.

[Gevarter, 83]. W. Gevarter, "Expert Systems: Limited but Powerful", Spectrum , Aug. 1983.

[Goguen, 68]. J. Goguen, "The Logic of Inexact Concepts", Synthese, vol. 19, pp. 325-373, 1968.

[Hart, 78]. P. Hart, R. Duda, and M. Einaudi, "A Computer Based Consultation System for Mineral Exploration", Tech Report, SRI International, Menlo Park, Calif., 1978.

[Hayes-Roth, 83]. F. Hayes-Roth, D. Waterman, and D. Lenat, (Eds.) Building Expert Systems, Addison-Wesley, Reading, MA., 1983.

[Klahr, 78]. P.Klahr, "Planning Techniques for Rule Acquisition in Deductive Question Answerings", in Pattern Directed Inference Systems, D. Waterman and F. Hayes-Roth, (Eds.), Academic Press, New York, 1978.

[Malhotra, 75]. A. Malhotra, "Knowledge-Based English Language System for Management Support: Analysis of Requirements", Proc. Fourth Int'l. Joint Conference on Artificial Intelligence, pp. 842-847, MIT AI Laboratory, Cambridge, MA, 1975.

[Martin, 77]. N. Martin, "Knowledge-Base Management for Experiment Planning in Molecular Genetics", Proc. Fifth Int'l. Joint Conf. on Artificial Intelligence, 1977.

[McCalla, et.al, 83]. G. McCalla, N. Cercone, "Approaches to Knowledge Representation", Computer, vol. 16, no. 10, Oct. 1983.

[McDermott, 81]. J. McDermott, B. Steele, "Extending a Knowledge Based System to Deal With Ad hoc Constraints", Proc. Seventh Int'l. Joint Conf. on Artificial Intelligence, 1981.

[Miller, 73]. P. Miller, "A Locally-Organized Parser for Spoken Input", Doctorial Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1973.

[Moses, 71]. J. Moses, " Symbolic Integration: The Stormy Decade", Comm. of ACM, vol. 14, no. 8, 1971.

[Mylopoulos, et.al. 75]. J. Mylopoulos, A. Borgida, P. Cohen, and N. Roussopoulos, "TORUS - A Natural Language Understanding System for Data Management", Proc. Fourth Int'l Joint Conf. on Artificial Intelligence, pp. 414-421, MIT AI Laboratory, Cambridge, MA, 1975.

[Nau, 83]. D. Nau, "Expert Computer Systems", Computer , vol. 16, no. 2, Feb. 1983.

[Nii, 78]. H. Nii, E. Feigenbaum, "Rule-Based Understanding of Signals", Pattern-Directed Inference Systems, D. Waterman and F. Hayes-Roth, (Eds.), Academic Presss, New York, 1978.

[Nilsson, 80]. N. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Co., Palo Alto, CA, 1980.

[Osborn, 79]. J. Osborn, "Managing the Data from Respiratory Measurements", Medical Instrumentation, vol. 13, no. 6, Nov. 1979.

[Pople, 77]. H. Pople, "The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning", Proc. Fifth Int'l. Joint Conf. Artificial Intelligence, 1977.

[Reddy, 73]. R. Reddy, R. Fennell, R. Neely, "The HEARSAY Speech Understanding System: An Example of the Recognition Process", Proc. Third Int'l. Joint Conf. Artificial Intelligence, pp 185-193, 1973.

[Reddy, 75]. R. Reddy, (Ed.) Speech Recognition: Invited Papers of IEEE Symposium, Academic Press, New York, N.Y., 1975.

[Shortliffe, 76]. E. Shortliffe, Computer-Based Medical Consultations: MYCIN, American Elsevier, New York, N.Y., 1976.

[Shortliffe, 75]. E. Shortliffe, B. Buchanan, "A Model of Inexact Reasoning in Medicine", Mathematical Biosciences, vol. 23, pp 351-379, 1975.

[Stallman, 77]. R. Stallman, G. Sussman, "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis", Artificial Intelligence, vol. 9, pp 135-196, 1977.

[Stefik, 77]. M. Stefik, N. Martin, "A Review of Knowledge Based Problem Solving as a Basis for a Genetics Experiment Design System", Computer Science Dept., Heuristic Programming Project memo HPP-75-5, Stanford Univ., Stanford, C.A., 1977.

[Tversky, 72]. A. Tversky, "Elimination by Aspects: a Theory of Choice", Psychological Review, vol. 79, pp 281-299, 1972.

[Weiss, 78]. S. Weiss, "A Model-Based Method for Computer-Aided Medical Decision-Making", Artifical Intelligence, vol. 11, no. 2, 1978.

[Wiederhold, 84]. G. Widerhold, "Knowledge and Database Management Systems", Software, vol. 1, no. 1, Jan. 1984,

[Winograd, 75]. T. Winograd, "Frame Representations and the Declarative/Procedural Controversy", in Representation and Understanding: Studies in Cognitive Science, D. Bobrow, A. Collins (Eds.), pp. 188-210, Academic Press, New York, NY, 1975.

[Winston, 77]. P. Winston, Artificial Intelligence, Addison-Wesley, Reading, MA, 1977.

[Zadeh, 75a]. L. Zadeh, "Fuzzy Logic and Approximate Reasoning", Synthese, pp 407-428, vol 30, 1975.

[Zadeh, 75b]. L. Zadeh, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I", Information Sciences, vol. 8, no. 3, pp. 199-249, 1975.

4.6

1. Report No. IN-82	2. Government Accession No. 183331	3. Recipient's Catalog No.
4. Title and Subtitle USL/NGT-19-010-900: KNOWLEDGE-BASED SYSTEMS: A PRELIMINARY SURVEY OF SELECTED ISSUES AND TECHNIQUES 104P.		5. Report Date May 31, 1984 6. Performing Organization Code
7. Author(s) SRINU KAVI		8. Performing Organization Report No.
9. Performing Organization Name and Address University of Southwestern Louisiana The Center for Advanced Computer Studies P.O. Box 44330 Lafayette, LA 70504-4330		10. Work Unit No. 11. Contract or Grant No. NGT-19-010-900
12. Sponsoring Agency Name and Address		13. Type of Report and Period Covered FINAL; 07/01/85 - 12/31/87 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Working Paper Series report surveying the state-of-the-art in knowledge-based systems (expert systems), including issues related to knowledge representation, knowledge bases, cognitive engine strategies, user interfaces for knowledge-based systems, and application considerations.

This report represents one of the 72 attachment reports to the University of Southwestern Louisiana's Final Report on NASA Grant NGT-19-010-900. Accordingly, appropriate care should be taken in using this report out of the context of the full Final Report.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Knowledge-Based Systems, Information Storage and Retrieval Systems	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	102	

*For sale by the National Technical Information Service, Springfield, Virginia 22161