KNOWLEDGE ENGINEERING TECHNIQUES AND TOOLS IN THE PROSPECTOR ENVIRONMENT

Technical Note 243

June 1981

By: René Reboh, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

SRI Project 5821,6415, and 8172

This report is a slightly revised version of a thesis submitted to the Department of Computer Science of the Linköping University, Sweden, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The research reported herein has been supported over a period of several years by the United States Geological Survey under Contract No. 14-08-0001-15985 and 14-08-001-17296 and by the National Science Foundation under Grant No. AER77-04499.

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-2046

SRI International

KNOWLEDGE ENGINEERING TECHNIQUES AND TOOLS IN THE PROSPECTOR ENVIRONMENT

Technical Note 243

June 1981

By: René Reboh, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

SRI Project 5821,6415, and 8172

This report is a slightly revised version of a thesis submitted to the Department of Computer Science of the Linköping University, Sweden, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The research reported herein has been supported over a period of several years by the United States Geological Survey under Contract No. 14-08-0001-15985 and 14-08-001-17296 and by the National Science Foundation under Grant No. AER77-04499.

SRI International®

ABSTRACT

Techniques and tools to assist in several phases of the knowledge-engineering process for developing an expert system are explored.

A sophisticated domain-independent network editor is described that uses knowledge about the representational and computational formalisms of the host consultation system to watch over the knowledge-engineering process and to give the knowledge engineer a convenient environment for developing, debugging, and maintaining the knowledge base.

We also illustrate how partial matching techniques can assist in maintaining the consistency of the knowledge base (in form and content) as it grows, and can support a variety of features that will enhance the interaction between the system and the user and make a knowledge-based consultation system behave more intelligently.

Although these techniques and features are illustrated in terms of the Prospector environment, it will be clear to the reader how these techniques can be applied in other environments.

CONTENTS

ACKNOWLEDGMENTS

The knowledge-engineering environment described in this report takes advantage of several features and programs that have been developed for the Prospector project as well as for other projects in SRI International's Artificial Intelligence Center (SRI-AIC). Several aspects of this work have been described in various SRI technical notes or final reports of the Prospector project, with contributions from many staff members of the SRI-AIC. The author presented a paper on the results described in Chapter V at the 1980 AAAI Conference [53].

Many people have contributed to the development of the Prospector system since the project began in 1975. Project leadership was shared by Peter E. Hart and Richard O. Duda, who also worked out most of the probabilistic computations used in the system. The author produced the overall system design and implementation. Kurt Konolige contributed to the development of the formal language for the external description of the knowledge base and wrote the original version of a parser for that language. John Gaschnig conducted performance evaluation experiments and devised several tools for performing that task. The semantic network package and the natural language interface (LIFER) were borrowed from the natural language group at SRI-AIC and were developed by Gary G. Hendrix and Jonathan Slocum, who also wrote the original LIFER grammar for interpreting volunteered information. Other contributors during various development phases include Phyllis Barrett and B. Michael Wilber. Finally, during the first year, Nils J. Nilsson and Georgia L. Sutherland made major system design contributions.

I am indebted to many of my colleagues at SRI-AIC and Linkoping University for valuable discussions and advice that guided the development of this work. In particular, I owe many thanks to Prof. Erik Sandewall at the University of Linkoping, Sweden, and to R. O. Duda, J. Gaschnig, P. E. Hart, and N. J. Nilsson at SRI.

Valuable comments on this or earlier versions of the manuscript were given by R. O. Duda and J. Gaschnig at SRI, S. Hagglund and A. Haraldsson at the University of Linkoping, and S. Lof of the Defense Research Center (FOA) in Sweden.

I INTRODUCTION

1. Background On Knowledge-Based Systems Technology

In recent years there has been a growing awareness among researchers in artificial intelligence (AI) [49] that if effective application programs are to be developed, general methods for performing tasks such as search, planning, and problem solving that apply to many different domains must be supplemented by extensive special knowledge about the particular domain of application [32]. A new generation of high-level languages (e.g., [57]) were developed as a step toward that goal. These languages introduced a variety of techniques for allowing domain knowledge to assist in controling the problem-solving process. However, the languages dealt only marginally with the problems of efficiency and representation that were still posed by many AI systems. Toward this end, general methods for representing large amounts of knowledge in forms that permit effective use of that knowledge in a variety of applications have gained renewed interest. Predicate calculus, frame systems, and production systems are three such general approches that have found wide application.* All three approaches primarily favor an explicit representation of knowledge in declarative form, separate from the program.

In predicate calculus, general-purpose, theorem-proving techniques [48] such as the resolution principle [55], are combined with pruning strategies [52] to deduce the consequences of knowledge represented uniformly as assertions.

In frame systems [43], highly structured objects are used to represent knowledge and many deductions are immediate properties of the representation (see Chapter II Section 2 and Chapter IV). In particular, the treatment of "inconsistencies" caused by exceptions to general rules is simplified in this formalism and individual objects automatically inherit properties from their membership in one or more classes of objects. Semantic networks [6, 39], units [65], and frame languages [5, 54] are examples of methods for representing structured objects. A discussion of the relations among these representations can be found in [49].

Production systems provide for a general computational formalism with three major components: a global database of assertions that represent the facts for a specific problem, production rules that read and write on this database, and a rule interpreter (also called the inference engine) that selects the rules to be applied [14, 45].

---

* A tutorial presentation of the principles underlying these methods can be found in [2], and an excellent introduction to these techniques and their application in AI can be found in [49].

Several flavors of production systems exist, including pattern-directed inference systems [72] and rule-based systems [73]. The rule-based systems approach, a generalization of the computational formalism of production systems, has been favored in most work aimed at developing knowledge-based expert systems. Several variants exist also in expert systems, differing in the method of knowledge representation, the chracteristics of the knowledge they use (e.g., subjective judgment), or in their mode of use (e.g., diagnosis). Working in specialized problem domains, these computer programs have achieved levels of performance that approach the skill of expert humans.

Table 1 shows a representative selection of well-known knowledge-based systems grouped according to their general function.

An important characteristic of knowledge-based systems (KBS) is that they are generally structured so that performance can be improved primarily by expanding or modifying the knowledge base (KB). Such systems display a rigid separation between the KB and the inference procedures used to draw conclusions from the KB. Production rules of the form <evidence> ---> <hypothesis>* constitute the primary formalism for encoding general knowledge about the domain. The rule interpreter can employ these rules in antecedent mode** or consequent mode.*** Advantages of the modular organization of rule-based systems include:

* Ease of modification and incremental development of the knowledge base.
* Ability (in principle) to produce an expert system in a different domain of application merely by substituting the knowledge base independently of the computational procedures. In addition, changing the reasoning mechanism while keeping the knowledge base fixed allows an expert system to be easily adapted to a different mode of use (i.e., consultation, instruction/education, autonomous operation).
* Requirement for relatively simple control mechanisms.
* Ability to "explain" the reasoning process in reaching a conclusion.

---

* Or <situation> ---> <action>

** Also called event-driven, data-driven, or forward-chaining mode.

*** Also called goal-driven, hypothesis-driven, or backward-chaining mode.

Table 1

GENERAL CATEGORIES of KNOWLEDGE-BASED SYSTEMS

Function	Domain	System	Reference
Search	Chemistry	DENDRAL	[23]
	Chemistry	SECHS	[78]
	Chemistry	SYNCHEM	[30]
Problem Solving and Planning	Circuit anal.	EL	[62]
	Genetics	MOLGEN	[64]
	Mechanics	MECHO	[11]
	Programming	PECOS	[3]
Diagnosis	Medicine	PIP	[50]
	Medicine	CASNET	[74]
	Medicine	INTERNIST	[51]
	Medicine	MYCIN	[61]
	Engineering	SACON	[4]
	Geology	PROSPECTOR	[20]
Machine Perception	Acoustics	HASP (SU/X)	[46]
	Imagery	ACRONYM	[7]
	Chemistry	CRYSALIS (SU/P)	[22]
	Elec. Warfare	MSIS	[27]
Simulation	Econometrics	OIL	[13]
CAI	Electronics	SOPHIE	[9]
	Medicine	GUIDON	[12]
Learning	Chemistry	Meta-DENDRAL	[10]
	Agriculture	INDUCE	[16]
	Mathematics	AM	[40]
Intelligent Assistants	Messages	RITA	[73]
	Business	COMEX	[63]
	Operations Res.	TESTBED	[56]
	Computer Systems Configurations	R1	[42]
Knowledge Acquisition	Diagnosis	TEIRESIAS	[15]
	Diagnosis	EMYCIN	[69]
	Diagnosis	EXPERT	[75]
System Building	---	ROSIE	[73]
	---	AGE	[47]

2. Background On Knowledge Acquisition Systems

Much work in artificial intelligence is concerned with knowledge acquisition (KA), and several approaches to the general problem of knowledge acquisition for expert systems have been investigated. For example, an extensive literature exists on the general topic of machine learning or induction, including the learning of rules for expert systems [10, 71, 76]. In addition, systems have been developed as generalizations of existing expert systems to simplify the development of a similar system in a new problem domain [47, 69, 75]. While such high-level languages for programming new knowledge-based systems promise to shorten system implementation time, they do not deal with the problem of shortening the time required for knowledge acquisition and knowledge-engineering. While the details of the inference procedures used to draw conclusions differ from one expert system to the other, the rules and other domain-related knowledge must be obtained by a time-consuming process of interviewing experts and encoding the collected information into an efficient representation that must reflect the experts' intentions.

Primarily for their own convenience, workers in the field of expert systems have developed a number of tools to assist in this process. The most ambitious approaches attempt to interact in natural language directly with the domain expert (DE) to acquire domain knowledge and new conceptual primitives without requiring at the same time that the user understand the details of their representation within the system [15, 44]. While this is the ultimate goal, we believe that its practical attainment is still a difficult research problem. We have chosen instead to develop a knowledge acquisition environement that requires the collaboration of a knowledge engineer (KE), but where the techniques and tools used supply intelligent support in critical phases of the knowledge acquisition process and require that the KE have only little training to use them.

3. Organization of this Report

Chapter II contains an overview of the Prospector environment in which the knowledge-engineering techniques and tools described in the remaining chapters were explored and developed. The representational formalisms as well as the inference procedures used in Prospector are described in the overview with just enough detail as necessary to facilitate the understanding of the features presented in subsequent chapters and to make this report relatively self-contained. Detailed descriptions and discussion of several aspects and features of the Prospector system can be found in reports and technical notes issued during the development of the system [18, 19, 20, 21].

Chapter III describes the knowledge-engineering process and some of the tools developed to assist in various phases of this process.

Chapter IV and V are devoted to the relatively detailed description of the two major components of Prospector's knowledge acquisition system (KAS), emphasizing the underlying general techniques employed.

Chapter IV describes a sophisticated domain-independent network editor (RENE) that uses knowledge about the Prospector environment and an automatic bookkeeping facility to assist in several phases of the development of an expert system. Chapter V describes a semantic network matcher and its use to assist in several aspects of the knowledge acquisition process as well as of the consultation system. Appendix A gives a brief account of our most recent experiences with KAS, illustrating in particular its use in domains other than geology.

Appendix B contains a computer transcript of a complete knowledge-engineering session to illustrate features of the network editor and its bookkeeping system, and Appendix C contains a description of the primitive commands of the network editor in primer form.

Finally, Appendix D contains a list of acronyms used throughout this report and a schematic key to Prospector model diagrams.

II OVERVIEW OF PROSPECTOR

1. Background and Domain of Expertise

A major activity in the field of economic geology is developing models of various classes of ore deposits. These models describe the distribution of ore minerals and their associated petrological and structural features. By organizing observations into a coherent pattern, such models give the exploration geologist a tool to interpret existing data and guide exploration decisions. The models also play a major role in regional resource evaluation by defining the regional and local characteristics favorable (or unfavorable) for specific types of ore deposits.

To develop a model requires both scientific understanding of the physical and chemical processes of ore deposition and geological judgment based on informed experience. Only very few types of deposits have been thoroughly explored and many have been discovered in just the past few years. The complex processes involved are therefore not yet well understood by earth scientists and much of what is known often depends upon empirical observations found only in the heads of a handful of highly specialized experts.

Ore deposit models encoded in the Prospector system have been developed with the collaboration of carefully selected American and Canadian economic geologists, who are regarded by their peers as being among the top authorities on each particular type of deposit considered. The primary goal of the Prospector system is to provide a field geologist (nonexpert) who is exploring a particular site with computer-based consultation to determine such things as

* Which model best fits the available field data
* Where the most favorable drilling sites are located
* What additional data would be most helpful in reaching firmer conclusions
* What is the basis for these conclusions and recommendations.

In performing these tasks, Prospector employs both logical and probabilistic reasoning procedures for interpreting subjective and uncertain evidence supplied by the user for a particular prospect, and for using the information encoded in formal ore deposit models designed by the domain expert. The rest of this chapter is devoted to the description of the different components and important features of the Prospector system.

2. Knowledge Base

The knowledge base (KB) of Prospector can be used in several ways. It can serve as a basis for a consultation session, a knowledge acquisition session, or simply as a repository of information to answer questions about the domain (i.e., for educational purposes). Construction of the KB is incremental and is continually extended (typically, over long periods), so more situations will be recognized. Some knowledge may be relevant to only one situation, whereas other information may be pertinent to several situations. For instance, a particular mineral, "pyrite," may be mentioned several times in the KB, but "pyrite dikes surrounded by dacite plugs" may occur only in an isolated situation. To facilitate many aspects of the development and use of the KB, it is divided into two main categories of knowledge that can be developed independently--a general-purpose knowledge base and a special-purpose knowledge base. The general-purpose knowledge base encodes as much as possible of background knowledge that is useful for several applications and situations of the domain. This KB comprises any general statements about the domain as well as the taxonomies describing the various concepts of the domain. The special-purpose knowledge base encodes statements that are relevant to some specific subset of the domain and contains primarily the inference networks (rules and other inference constructs, see below) in which these statements participate. In the next few sections we illustrate how statements, taxonomies, and inference networks are encoded in Prospector. Only those aspects of the representation that will help the reader understand the examples and features presented in subsequent chapters will be described.

2.1. Taxonomies

Taxonomies are represented as hierarchical tree structures where the nodes are simple concepts of the domain connected by arcs indicating the element (e) and subset (s) relationships between these concepts. Because the knowledge of whether or not an item belongs to a given set is essential in question answering and fact retrieval, the taxonomy itself often provides a natural and concise expression of portions of the information about a task domain. Furthermore, the taxonomy organization allows any property that is characteristic of all members of a given set to be described at the set level and need not be repeated in the encoding of each individual set member (inheritance of properties).

Each node X in the hierarchical structure is said to be a restriction of its parent nodes or of any node occurring on a chain of outgoing "e" and "s" arcs from X. Similarly, X is a generalization of any descendant node occurring on any chain of incoming "e" and "s" arcs to it. Two references X and Y to entries in the taxonomy are said to be compatible if they are synonyms* or if X is either a restriction or generalization of Y (i.e., occur on the same e-s chain).

---

* Synonyms correspond to a unique node in the taxonomy representation.

Many sibling subsets described in taxonomies are disjoint. For a more precise network encoding of taxonomies, the standard set theory notions of set membership and set inclusion, which are expressed by "e" and "s" arcs, are supplemented by the more restrictive concepts of disjoint subsets (ds) and distinct elements (ds). A "ds" arc from a node X to a node Z indicates that X is a subset of Z and that X is disjoint from any other set Y with an outgoing "ds" arc to Z. Similarly, arcs labeled "de" (for "distinct element") indicate that each of two or more nodes denotes a different element of a set.

Distinctness and disjointness properties may be propagated through the taxonomy network. If A has an outgoing "de" arc to S1, and B has an outgoing "de" arc to S2, and there are unbroken paths of "ds" arcs from both S1 and S2 to some common superset S0, then S1 and S2 are disjoint, and A and B are distinct. In fact, every element of S1 is distinct from every element of S2. Using "ds" and "de" arcs increases the representation power of the taxonomy. For example, the "ds" arcs in Figure 1 emanating from the nodes labeled MATERIALS, FORMS, AGES, and so on, indicate that the set of MATERIALS, the set of FORMS, etc. are disjoint subsets of the set UNIVERSAL. This means in particular that the subtrees below MATERIALS, FORMS, and so on, are disjoint. Thus, given (the label of) a node contained in any of these subtrees, it would be a simple matter in this representation to tell if it is a physical form, an age, a mineral, or a rock type (see delineations below). Information about nonintersection and nonequivalence can be encoded by other means, but the "de" and "ds" arcs allow much of this information to be encoded for the price of the hierarchical information alone, without additional structure.

Examples of taxonomies in the KB of Prospector include rock types, minerals, physical forms, and geological ages (Figure 1).

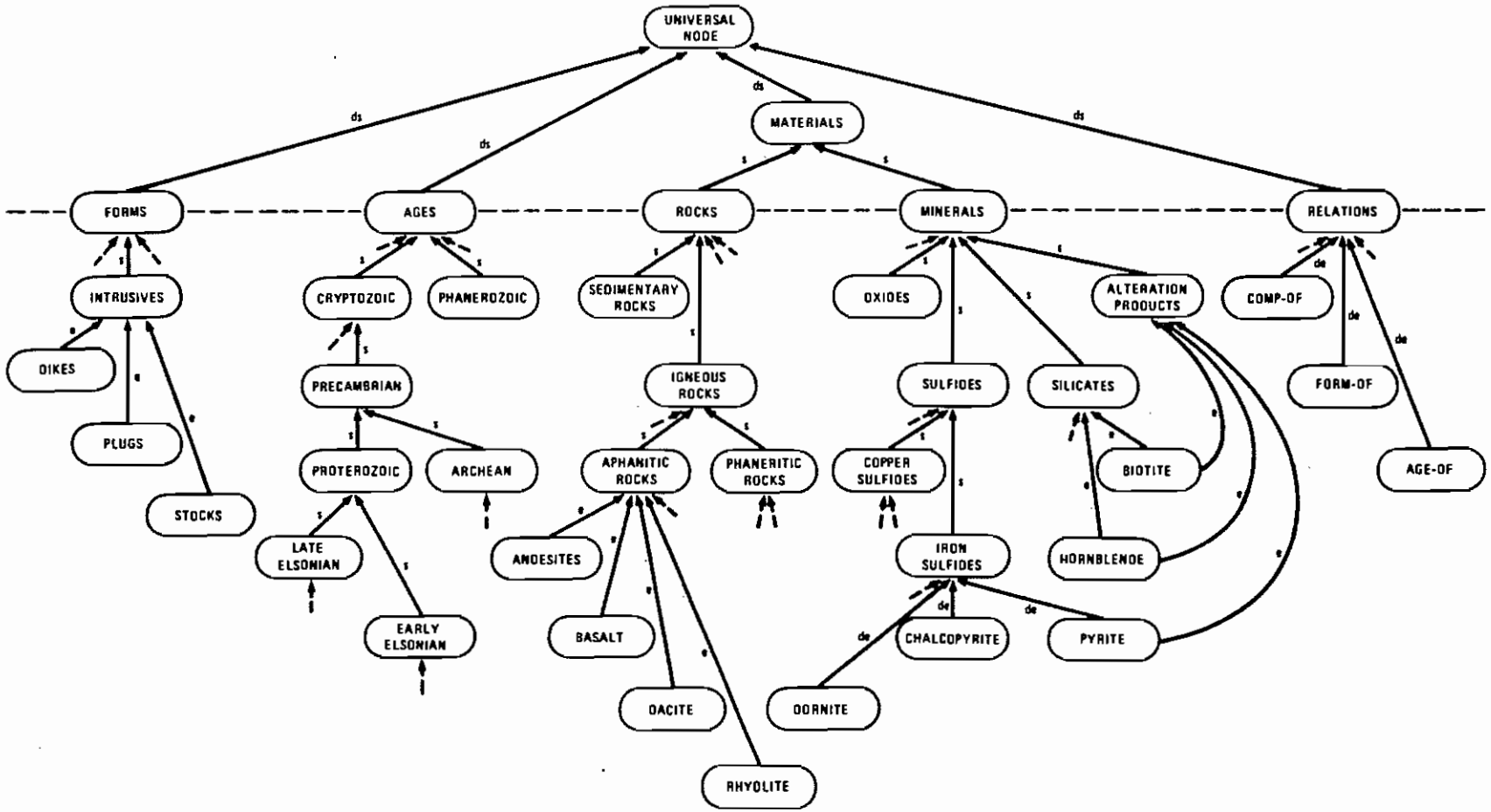


Figure 1 Examples of Taxonomies in Prospector

2.2. Statements

The basic concept of a network as a collection of nodes and arcs can be extended by partitioning groups of nodes and arcs and allowing them to be bundled into units. These units can then correspond to nodes in a "higher level" network (such as the inference networks described below). This partitioning adds a new dimension to the organizational and expressive power of network representation.

Partitioned semantic networks [39] are used to encode statements in the knowledge base. Each statement is represented by a structured object [49] called a space, where the semantic representation of the statement is stored in terms of primitive relations and entries in the various taxonomies of the domain of application (in this case geology). Let us illustrate this with simple examples from Prospector. The statement

"a rhyolite plug is present"

is represented in Figure 2 as a small network that makes the following three assertions:

(a_1) There exists a physical entity E_1.
(a_2) The composition of E_1 is "rhyolite."
(a_3) The form of E_1 is "plug."

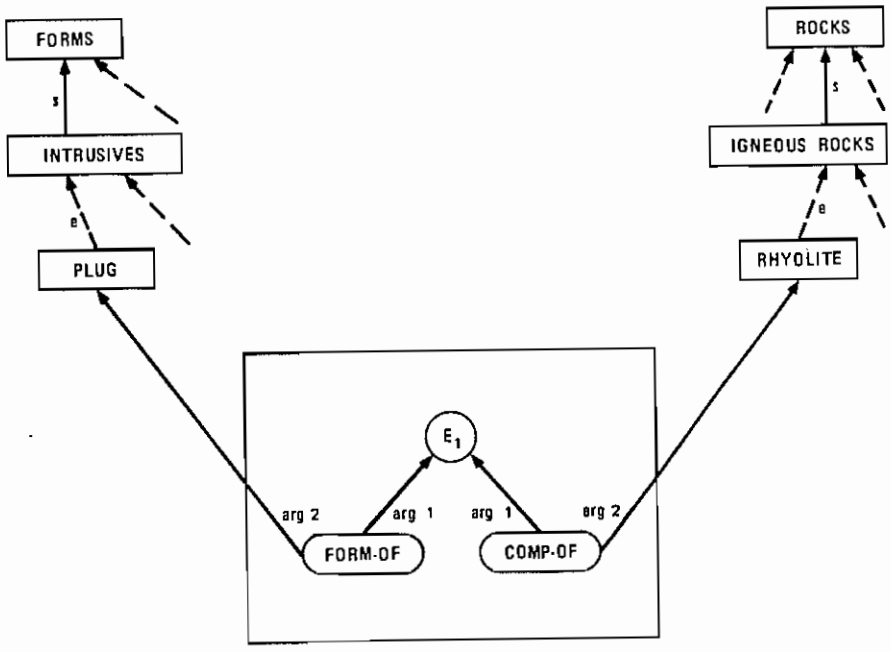


Figure 2 Semantic Network Representation of "a rhyolite plug is present"

In general, an assertion corresponds to a node inside the space (delimited by the rectangle in Figure 2) that constitutes the semantic representation of the statement. Because "rhyolite" and "plug" can be referred to from other statements of the KB, they are not included in that space, but appear instead as entries in the taxonomy of "rocks" and the taxonomy of "forms," respectively. We will call such references external references of a space. Most frequently, external references are entries in the taxonomy, but may also be disjunctions or conjunctions of such entries as well as other concepts that are included in the semantic representation of any other statement in the KB (examples of such situations will be given in Chapter V).

In addition to physical entities, a variety of other concepts such as places (locations) and geological processes are described in the KB. Attributes associated with a concept appear in the semantic representation as relations of two or more arguments. Composition and form (Figure 2) are two common attributes of physical entities. Examples of binary relations used in Prospector are:

AGE-OF
COMP-OF (composition)
FORM-OF
GRAIN-SIZE-OF
LOC-OF (location)
SIZE-OF
TEXTURE-OF.

N-ary relations include:

ALTERED-TO (example in Figure 3)
COMPONENTS-OF (e.g., the components of an entity E_1 are E_2 and E_3)
CONTAINED-IN (e.g. E_5 and E_6 are contained in E_7)
RELATIVE-AGE-OF (e.g. E_8 is "younger" than E_9).

The first argument of the relation refers always to the concept being described; the other arguments are values of attributes associated with that concept. Most frequently, these attribute values are external references but can also be other concepts included in the statement being described. For instance, the statement

"partially biotized hornblende"

describes an alteration process entailing two entities, the original substance (hornblende) and the substance to which it becomes altered (biotite). The process has been completed to some degree (partial). The semantic representation of this statement corresponds to a network (Figure 3) that makes the following seven assertions:

(a_1) There exists a physical entity E_1.
(a_2) The composition of E_1 is "hornblende."
(a_3) There exists a physical entity E_2.
(a_4) The composition of E_2 is "biotite."
(a_5) There exists a process P_1.
(a_6) In the process P_1, E_1 is being altered to E_2.
(a_7) In P_1, the degree of alteration is "partial."

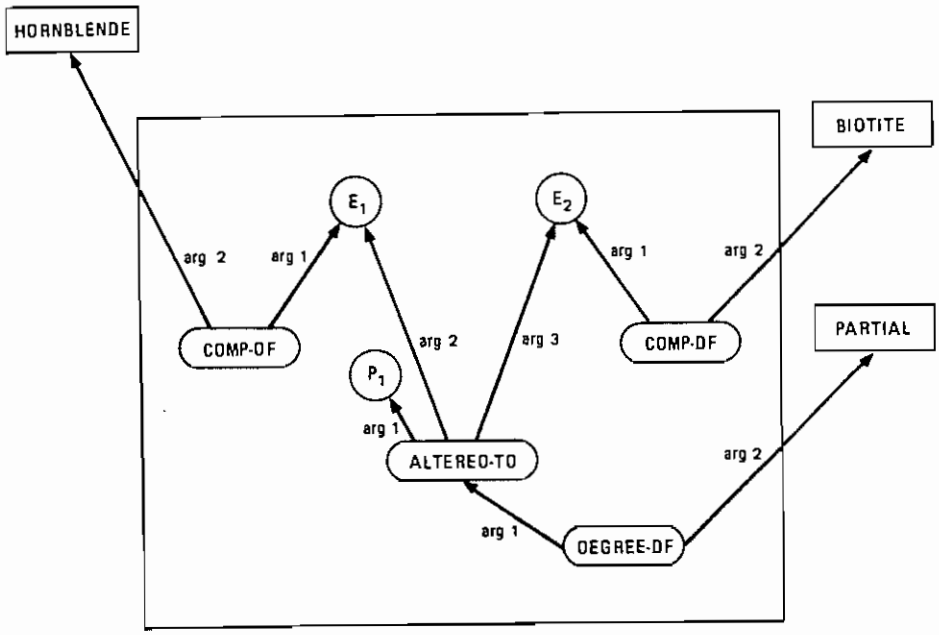


Figure 3 Semantic Network Representation of "partially biotized hornblende"

Some attributes may be much more specialized than others, some may appear only once in the entire KB. While such attributes can be assigned new primitive relations, the additional effort and complexity introduced are not always motivated. A universal catchall relation, ENTITY-PROPERTY, encodes all such isolated attributes. In such cases, the semantic representation says merely that some entity E has an "entity-property" whose value is the attribute. Examples from Prospector of such attributes include: "fresh looking" , "youngest major stock", "alteration halo", "shallow water shelf type sediments deposited under stable tectonic conditions," and so on. As more concepts are introduced and such attributes appear in more than one statement, further refinement of the semantic representation may become justified.

2.3. Delineations

A delineation, associated with each of the primitive relations, is a pattern that describes the number and type of the relations arguments. For instance, the delineation for the COMP-OF relation specifies that the first argument is a "physical entity" and the second argument should be a "rock type" or a "mineral." During a consultation, when the user volunteers an observation such as "a rhyolite plug is present," a corresponding semantic representation is constructed (Figure 2). The delineation associated with the primitive relations, together with the hierarchical information contained in the taxonomies, ensure that the COMP-OF relation will be associated with "rhyolite" while the FORM-OF relation will be associated with "plug." Delineations are particularly useful during the model-building phase where the KE may be editing elements of the semantic representation of statements and "type checking" of the arcs and nodes involved is needed.

2.4. Inference Networks

Ore deposit models are represented in Prospector by inference networks that relate field observations or evidence to the hypotheses that are the important constituents of the models. An inference network is equivalent to a collection of rules of plausible inference called inference rules. In general, an inference rule has the form

IF E THEN (to degree LS, LN) H.

The rule is interpreted to mean "The observed evidence E suggests (to some degree) the hypothesis H." A probability of truth is associated with every observation and hypothesis, and the inference rules specify how the probability that the hypothesis (right-hand side) is true is changed by the observation of evidence (left-hand side). The two parameters LS and LN establish the "strength" of the rule and specify how the probability of H is to be updated given that of E. In general, we need to be able to say both how encouraging it is to find the evidence E present, and how discouraging it is to find it absent. The two numbers, LS and LN, thus specify the sufficiency and the necessity measures, respectively, and must be supplied by the DE for each rule in the inference network.

Different pieces of evidence can also be combined logically to form a single, compound piece of evidence. The simpler elements are combined by means of the primitive operations of conjunction (AND), disjunction (OR), and complementation (NOT).

Rules can interconnect in various ways--through "chains" where the hypothesis for one rule is the evidence for another, through several pieces of evidence bearing on the same hypothesis, and through the same piece of evidence bearing on several different hypotheses. Figure 4 shows a portion of the inference network encoding a Prospector model for porphyry copper deposits.

The way information about several pieces of evidence is combined to reach a conclusion about any hypothesis and how the two parameters LS and LN change the probability are described in Section 3 below.

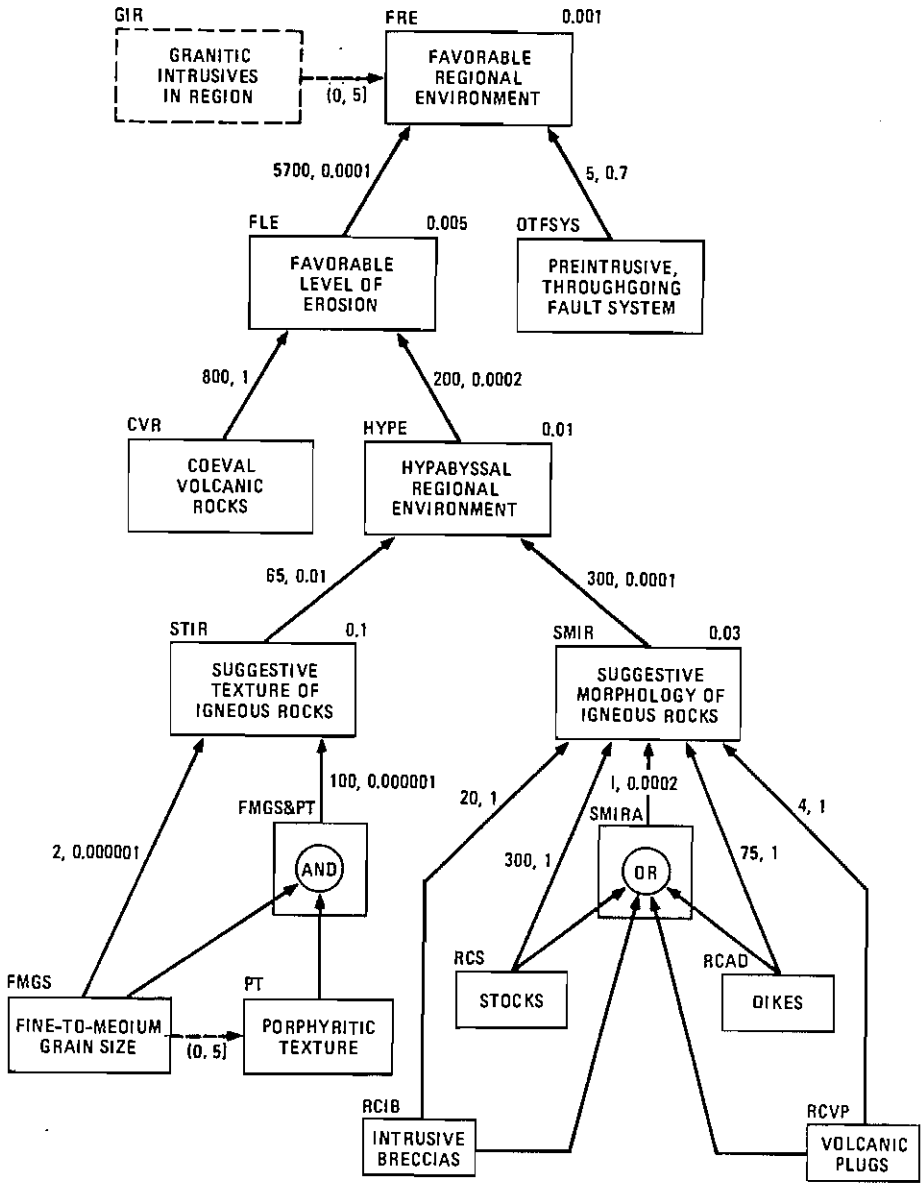


Figure 4 Sample Inference Network from a Prospector model

2.5. Size of the KB

Prospector's KB now contains fifteen models with over 1000 rules and 1500 spaces. The taxonomies shared by all models contain over 1000 entries.

To overcome virtual address space limitations, a "swappable" version of Prospector allows the KB (primarily the inference networks and taxonomies) to reside in external storage, allowing Prospector to consider a large number of models, but rendering slower instead some aspects of the consultation (such as volunteering observations).

3. Inference Procedures

To deal with the uncertainty of user observations, Prospector uses an inference mechanism based primarily on subjective probability theory [17, 25] supplemented with certainty theory [60] and fuzzy sets [79].

3.1. Subjective Probabilities and Bayes's Rule for Certain Evidence

A probability is associated with every statement in the KB (corresponding to a node in the inference network), measuring the degree to which the statement is believed to be true. When engaged in consultation about a particular prospect, Prospector records the specific geological evidence as it is furnished by the user and the values of the probabilities are updated. The basis for the procedure to propagate a probability from evidence E to hypothesis H is an elementary theorem of probability theory called Bayes's rule.

The Bayesian method assumes that before any information has been obtained from the user, every statement S has some prior probability P(S). As evidence is acquired during the consultation, a posterior probability corresponds to the updated probability of S. If E' denotes all the evidence accumulated up to some point in the consultation, then the posterior probability P(S|E') denotes the current probability of S given the evidence E'. The prior probabilities are generally supplied by the DE at the time the model is constructed, but can also, in some cases, be computed from the prior probabilities assigned to related spaces (e.g., logical components).

For this overview, we will use the odds likelihood* form of Bayes's rule:

$$O(H|E) = LS * O(H) \qquad (1)$$

This form relates three quantities involving an evidence E and the hypothesis H:

O(H)	prior odds on the hypothesis
O(H|E)	posterior odds on the hypothesis (given that the evidence E is observed to be present)
LS	sufficiency measure of the rule

The quantity LS has a standard interpretation in statistics and is called the likelihood ratio. It is defined by the formula

---

* The odds O and probability P for any situation are related by O = P/(1-P) so that odds and probabilities are completely equivalent.

$$LS = \frac{P(E|H)}{P(E|\neg H)}.$$

LS (which is never negative) is greater than one if E is more associated with H than with ¬H, is less than one if E is more associated with ¬H than with H, and is equal to one if E and H are statistically independent.

Thus, Bayes's rule specifies that the observation of E changes the odds on H by the factor LS. Large values of LS mean that E is favorable for H. If LS is infinitely large, then E is logically sufficient for concluding H. Small values of LS mean that E is unfavorable for H. If LS=0, then H is false when E is true, so that ¬E is logically necessary for concluding H. If LS=1, then the observation of E has no effect on the belief in H.

Analogous formulas and remarks hold for the case in which the evidence E is observed to be definitely absent, in which case Bayes's rule has the form

$$O(H|\neg E) = LN * O(H)$$

where the likelihood ratio LN corresponds to the necessity measure of a rule and is defined by

$$LN = \frac{P(\neg E|H)}{P(\neg E|\neg H)}.$$

The above formulas clearly show that LS and LN must be nonnegative, and that because it is not possible for both E and ¬E to be favorable for H, or both to be unfavorable, only the following cases can occur:

$$LS > 1 \text{ and } LN < 1$$
$$LS < 1 \text{ and } LN > 1$$
$$LS = LN = 1.$$

However, numerical values for both the prior probabilities and the likelihood ratios are obtained from the DE (who provided the rules), and these subjectively obtained values do not always satisfy these simple constraints. In particular, it is not unusual for an expert to assert that the observation of a particular evidence E is important, but that its absence has absolutely no significance. This implies that $LS \neq 1$ and $LN = 1$, which is not one of the three cases listed above. This and other practical problems are treated below.

3.2. Uncertain Evidence

In actual practice the user is often unable to observe either the definite presence or absence of the evidence. Typically, the user is prepared only to indicate a degree of confidence that the evidence sought is actually present. In this case, Prospector uses a formula that effectively interpolates (piecewise linear interpolation) between the two extreme cases of perfect certainty.

Let E′ denote the observations that cause the user to suspect the presence of the evidence E. The posterior probability P(H|E′) is thus

somewhere between $P(H|\neg E)$ and $P(H|E)$. In [17] it is shown that under certain (reasonable) assumptions, $P(H|E')$ is a linear function of $P(E|E')$, with $P(H|E') = P(H|\neg E)$ when $P(E|E') = 0$ and $P(H|E') = P(H|E)$ when $P(E|E') = 1$.

Because the probability values $P(H|E)$, $P(H|\neg E)$, $P(H)$ and $P(E)$ are all obtained from the DE's subjective estimates, they may often be inconsistent with the theoretically expected values. In particular, we must make certain that when nothing is known about E, i.e., when $P(E|E') = P(E)$, the interpolation function should leave H at its prior, yielding the theoretically expected value $P(H|E') = P(H)$. For example, if the DE says that $LS > 1$ and $LN = 1$, then $P(H|\neg E) = P(H)$, and thus when $P(E|E') = P(E)$ (which is positive in nontrivial situations), the interpolation function must yield $P(H|E') > P(H)$.

$P(H|E')$ is therefore chosen to be a piecewise linear function of $P(E|E')$ so that the desired values for $P(H|E')$ are obtained at the three fixed points $P(E|E') = 0$, $P(E)$, and 1. The resulting function is shown graphically in Figure 5 and can be expressed analytically as follows:

$$P(H|E') = \begin{cases} P(H|\neg E) + \dfrac{P(H) - P(H|\neg E)}{P(E)} P(E|E'), & 0 \leq P(E|E') < P(E) \\ P(H) + \dfrac{P(H|E) - P(H)}{1 - P(E)} [P(E|E') - P(E)]. & P(E) \leq P(E|E') \leq 1 \end{cases} \quad (2)$$

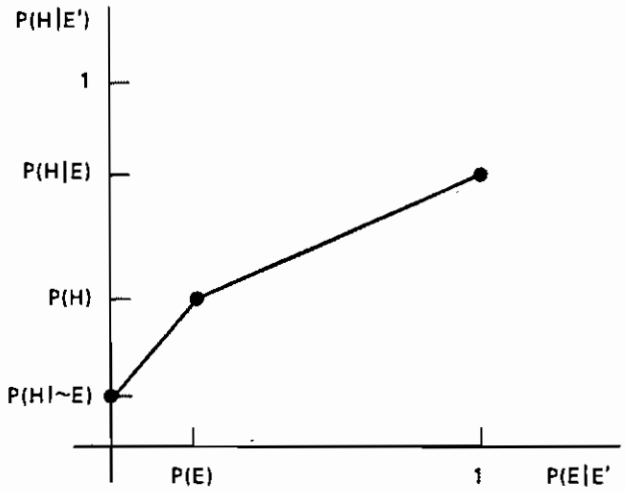


Figure 5 Consequent Probability as a Function of the Antecedent Probability

This formula, although not justified by traditional probability theory, has proven to be an effective practical procedure for treating

uncertain evidence. In particular, if LS > 1 and LN = 1, then P(H|E′) is unchanged if P(E|E′) is less than the prior probability P(E), but P(H|E′) increases with P(E|E′) if P(E|E′) is greater than P(E). This result captures the intent of the DE who gives these likelihood ratios, and eliminates the problem of error amplification during propagation because of the possible inconsistencies in parameters subjectively obtained from the DE.

3.3. Use of Certainty Measure in Communicating with the User

During a consultation, Prospector communicates with the user in terms of certainty measures rather than probabilities. The basic problem is that the user must be allowed to express his uncertainty about an evidence E, independently of the prior probability P(E) that was specified by the DE. For instance, if the user wants to express that E is present to some degree, he must specify a probability P(E|E′) that is greater than P(E), and thus must know P(E) so that he can choose a value of P(E|E′) relative to the reference point P(E). Using probability estimates to express user uncertainty is particularly unsatisfactory when rare events are involved because different users will assign to the same event values of P(E|E′) whose ratios are orders of magnitude apart.

Instead of asking for P(E|E′), Prospector asks for the certainty measure C(E|E′). The certainty measure can be viewed as the posterior probability normalized with respect to the prior probability. It is arbitrarily scaled from -5 to 5, and corresponds to the piecewise linear function of P(E|E′) (Figure 6), chosen so that P = 0 corresponds to C = -5, P = P(E) corresponds to C = 0, and P = 1 corresponds to C = +5.

$$C(E|E') = \begin{cases} 5\,\dfrac{P(E|E') - P(E)}{1 - P(E)} & \text{if } P(E|E') > P(E) \\ 5\,\dfrac{P(E|E') - P(E)}{P(E)} & \text{otherwise.} \end{cases}$$

This function is also used by Prospector when communicating to the user the certainties for conclusions. Rather than seeing the actual probabilities associated with the conclusions, the user is thus shown only how these probabilities have changed from their prior values. Since the certainty is a piecewise linear function of the probability and P(H|E′) is a piecewise linear function of P(E|E′), a piecewise linear function C(H|E′) of C(E|E′) can be used for combining the uncertainty of the rule E--->H and the uncertainty of the evidence. This provides the DE with an alternative way to specify rule strengths. Rather than having to specify the likelihood ratios LS and LN, the DE often finds it more convenient to specify the effect of the rule by giving C(H|E) and C(H|~E).

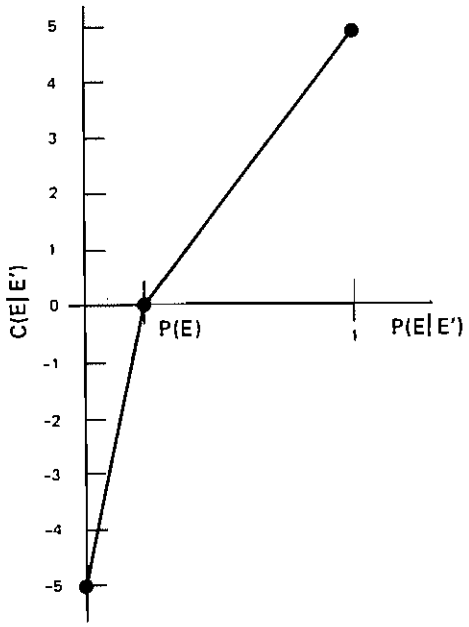


Figure 6 Relation between Certainty and Probability

3.4. Quantitative Rules

Prospector allows antecedents of rules to contain variables enabling the strength of the rules to depend on the values of those variables. Many of the rules in Prospector concern quantitative data, such as ages of rocks, chemical concentrations, and grain sizes of crystals. Prospector treats variable-strength rules as a simple generalization of the single-strength case above. For example, in the case of certain evidence, the posterior odds on the hypothesis H can be obtained from

$$O(H|x) = L(x) * O(H) ,$$

where the likelihood ratio L(x) depends on the value of the variable x and is defined by

$$L(x) = \frac{p(x|H)}{p(x|\sim H)} ,$$

where p is the probability density function for x. If x is not known with certainty, [19] shows in detail how a generalization of the interpolation procedure given by Eq. (2) is used to compute P(H|E').

For quantitative rules, the DE must specify a prior distribution function of the hypothesis over the values of the variable x (usually a few intervals) as well as a rule strength in each of the intervals.

Quantitative rules allow a clear distinction to be made between the numerical value of a quantity and the user's certainty regarding that

value. During a consultation, the user will be asked to specify an interval (a,b] within which x is believed to lie, and a certainty C associated with that belief. Prospector will interpolate appropriately when the interval given by the user overlaps several or parts of the intervals that were specified by the DE when the prior distribution function was given.

3.5. Combinations of Evidence

Consider the situation in which N pieces of evidence E_1, E_2, ..., E_N, all bear on the hypothesis H. While there are infinitely many ways in which the effects of the evidence might combine, it is often possible to approximate reality by one of the following idealized situations: independent evidences or logical combinations of evidences. We consider each in turn.

3.5.1. Independent Evidence

For the case in which each piece of evidence E_i has some effect on the belief in H, the posterior odds $O(H|E')$ is obtained by the equation

$$O(H|E') = \left[\prod_{i=1}^{N} L'_i \right] O(H), \qquad (3)$$

where

$$L'_i = \frac{O(H|E'_i)}{O(H)}.$$

L'_i is the effective likelihood ratio for E_i and the posterior odds $O(H|E'_i)$ are obtained from Eq. (2). In [17] it is shown that this procedure corresponds to the case where the E_i are statistically independent (under both the hypothesis H and the hypothesis ~H), when the evidence is known to be true or false. (In which case $L'_i = LS_i$ if $P(E_i|E'_i) = 1$ and $L'_i = LN_i$ if $P(E_i|E'_i) = 0$). When nothing is known about the evidence, this procedure leaves the posterior odds at their prior value (since $L'_i = 1$ if $P(E_i|E'_i) = P(E_i)$).

3.5.2. Logical Combination of Evidence

The antecedent of some rules is a logical combination of several pieces of evidence, such as

$$E = E_1 \text{ OR } (E_2 \text{ AND } \sim E_3).$$

Simple formulas from fuzzy set theory [79] are used to compute $P(E|E')$ from the probabilities of the individual components as follows:

Conjunction: $E = E_1$ AND E_2 AND ... AND E_N.

$$P(E|E') = \min_i \{P(E_i|E')\} \tag{4}$$

Disjunction: $E = E_1$ OR E_2 OR ... OR E_N.

$$P(E|E') = \max_i \{P(E_i|E')\} \tag{5}$$

Negation: $E = \neg E_1$

$$P(E|E') = 1 - P(E_1|E'). \tag{6}$$

In most situations, the components of a conjunction or disjunction are not statistically independent and therefore the above formulas have worked satisfactorily, the only disadvantage is the insensitivity of $P(E|E')$ to any but one of the components.

3.5.3. General Combinations

The ability to form conjunctive, disjunctive, logical, and independent combinations of evidence allows a flexible "language" for expressing the DE's judgment as to the ways that different pieces of evidence combine to bear on an hypothesis. In most situations, simple conjunctive, disjunctive, or independent combinations provide for the correct combining mechanism, but combinations of several of these basic mechanisms may sometimes be desirable. Consider, for instance, a case in which either E_1 or E_2 implies H. However, suppose that E_1 is stronger evidence than E_2, $(LS_1 > LS_2)$, and that if E_1 is present then the presence or absence of E_2 is irrelevant. This combination can be achieved by introducing two new hypotheses H_1 and H_2 for the rules

$$E_1 \text{---}>H_1 \text{ and } E_2 \text{---}>H_2$$

and by taking H to be the disjunction of H_1 and H_2.

3.6. Propagation of Probabilities

In general, any hypothesis H is either a logical combination of other statements or is the consequent of one or more inference rules. The present probability of H is obtained either by Eqs. (4), (5), and (6) in the former case, or by Eq. (3) in the latter case.

Equation (3) is a generalization of Bayes's rule. Whenever the probability of an antecedent for H changes, the corresponding effective likelihood ratio changes, and the probability of H is updated by reevaluating Eq. (3). If H is the antecedent for other rules, or a logical element of a higher level hypothesis, these computations are repeated to update the probabilities of the new hypotheses. Thus, by repeated use of these formulas, the effects of acquiring new evidence are propagated throughout the inference network.

4. Consultation System and Control Mechanism

The user communicates with Prospector through an executive that supports mixed-initiative interaction [8]. When Prospector is in control, the control strategy (the Executive) must, in principle, make two important decisions:

* Which of the top hypotheses should be worked on.

* Which question should the user be asked next.

In Chapter V Subsection 2.2.2.1.2, we illustrate how the user can influence the selection of the top hypothesis by volunteering observations. In addition, the user has the option of taking control any time during the consultation by typing any of several commands provided in Prospector instead of answering the question asked by the system. Many commands allow the user to obtain various kinds of information during a consultation. For instance, the user can request a more detailed rephrasing of the question if it appears to be ambiguous to him, interrogate Prospector about its reasons for asking the question, or invoke the explanation system to obtain a summary of the state and reasons for the present evaluation of the prospect. Other commands permit the user to influence the course of the consultation by changing previous answers or requesting that Prospector investigate a different hypothesis than that being selected.

The method for selecting an appropriate question to ask is basically a depth-first (left to right) strategy (for traversing the inference network) combined with several selection and stopping criteria that are evaluated along the path. For instance, if in order to establish an hypothesis H, a set of rules is about to be applied, the rules are ranked with respect to their ability to change the odds on H. Several factors are considered and a rule may be selected either because it has the highest potential for increasing the odds on H or in ruling it out. The potential effects of the rules that have not yet been applied are also computed, and Prospector does an "extreme-case" analysis to assess whether they will be able to make any significant change in the odds of H. Ineffective questions can be avoided if it is found that to pursue and establish H is not worthwhile.

Prospector formulates its questions taking into consideration the particular circumstances that have occurred up to that point in the consultation (see Chapter V Subsection 2.2.2.2). Questions are also formulated differently, depending upon the type of answer expected (i.e., a yes/no answer, a certainty measure, a value range if the evidence is involved in a quantitative rule, and so on). If the question is answered as expected, its effects (primarily the updating of probabilities) are propagated through the inference network and a new evidence is chosen to ask the user.

During model design, several constructs are available that allow the KE to insert control information into the inference network. Such constructs, when encountered during a consultation, impose additional constraints on how the control strategy should select the next question to ask the user. They play an important human-engineering role in the

design of the model because they allow the DE to override, when appropriate, the normal effects of the inference procedures on the question selection criteria. The most important constructs are discussed below.

4.1. Contexts

The context construct primarily ensures that two or more hypotheses will be established in a particular order, overriding normal selection strategies. Often it does not make sense to ask the user about a particular piece of evidence until a proper context for the question has been established. H_1 is said to be a context for H_2 if H_1 must be established before H_2. (In the inference network diagrams this is indicated by a dashed arc from H_1 to H_2.) This concept is further generalized by allowing the DE to specify an interval of certainty measures that indicates how the context is to be established. When no interval is specified the default interval (0,5] is assumed and means that H_1 is established if $C(H_1|E')$ is positive. An interval of [-1,1] would mean that H_2 should be established only if H_1 is uncertain, and an interval of [-5,0) means that H_2 should not be attempted unless H_1 has been established to be false. If the purpose is solely to force H_1 to be established before H_2 the interval [-5,5] is used.

4.2. Blocks

The DE can specify a given node in the inference network to be a block. Once such a node is encountered during a consultation, it must be completely investigated before another line of reasoning is pursued by the control strategy (unless the user specifies otherwise). This feature keeps related questions grouped together and prevents confusing switches between topics during the consultation.

4.3. Iterative Use of Prospector

The DE can specify a given set of nodes in the inference network to be submodels. This means that the submodels participate in a subclassification problem and that Prospector will go through as many iterations as needed by the user to establish the degree of fit of the submodels. Each iteration corresponds to a recursive call of Prospector with the submodels being the top-level hypotheses. When control is returned from this procedure (after the submodels are investigated or at the user's instructions), the probabilities associated with the submodels can be combined as desired to be propagated through the higher level inference network.

This mechanism is used in Prospector primarily to handle zones of alteration and mineralization in some exploration models where the recognition of several characteristic zones is very important for establishing the degree of match of the model. Each different type of zone is described by an inference network for a zone submodel. During the consultation, the user is asked to subdivide the prospect into distinct zones and Prospector is called recursively for each zone

indicated by the user to establish the best match with the encoded zone submodels. Any answers or volunteered observations are naturally "local" in each iteration and are associated only with the particular zone identified by the user.

4.4. Other Modes of Use of Prospector

In addition to the interactive mode providing for on-line consultation, Prospector allows two other modes of operation:

* Batch mode: Prospector is used primarily in this mode for processing data from questionnaires for off-line prospect evaluation or performance analysis of models during the design phase (see Chapter III Subsection 3.3). It is also a convenient feature for many applications that require large numbers of similar consultations [28].

* Graphic-IO: In this mode most of the input data are in the form of digital contour maps. This mode is used primarily for drilling-site-selection models and the maps describe the distribution of evidence or properties that are relevant for the evaluation of drilling sites and corresponding to nodes in the inference network. The maps are obtained interactively from the user and saved on external data files. For each set of such maps that the user wants to use as input data for Prospector, the relevant paths from the corresponding evidences are retrieved from the inference network and "compiled" into an efficient piece of code for computing the probability of the top-level hypothesis, given the values of the evidences at any point of the maps (for each point, the results are equivalent to those obtained in a regular consultation if the values of the evidences were volunteered and propagated in consequent mode through the inference network). Typically, the maps are divided into a grid of 128 x 128 sample points, and the compiled inference network is thus executed 16384 times. The result is a color-coded favorability map of the prospect for each combination of maps considered [19].

III KAS--KNOWLEDGE ACQUISITION SYSTEM

1. Knowledge Acquisition Problem

The performance of a knowledge-based system such as Prospector is directly dependent upon the amount, quality, and organization of the knowledge it contains. Therefore, a major portion of the effort devoted to developing such systems goes into formalizing and encoding expert knowledge about the domain of application. While Prospector's knowledge representation and inference techniques described in Chapter II supply a framework in which the performance of the system can be extended in a conceptually straightforward fashion (by adding more rules and models), the task of constructing, modifying, testing, and maintaining the KB tends to be very time consuming, making the efficiency of the knowledge acquisition process the crucial bottleneck in the overall development of an expert system. Two major characteristics of this task are that it involves large representation structures and that it is performed over an extended period of time (sometimes involving several domain experts and knowledge engineers). For example, the inference networks for typical ore deposit models in Prospector contain more than 150 rules and over 200 nodes. The fifteen current models took more than four years to develop with the collaboration of a dozen experts. Developing each model involves several stages of elaboration, refinement, and revision. In addition, as new models are developed, old ones must often be revised because of new developments in the field of expertise and a certain amount of interaction and overlap is often present among models.

We are restricting our interest to the problem of acquiring knowledge from experts for judgmental, rule-based consultation systems and have chosen to develop knowledge acquisition tools that can require the collaboration of two types of people:

* One or more domain experts (DE) who are authorities on the subset of the domain knowledge (model) which is to be encoded. In Prospector, these were exploration geologists who are authorities on the type of ore deposits considered.
* One or more knowledge engineers (KE) who are computer scientists and can translate the DE's intentions into the computer representation used in the knowledge-based system.

In addition to interviewing the DE and providing guidance in developing an overall structure for the model, the KE must also be aware of potential deficiencies in representations of models, detect their existence when present, and cooperate with the DE to correct them. The KE may therefore be expected to understand the operation of the consultation system in which the model is to be embedded.

2. Knowledge-Engineering Process: Methodology

In developing exploration models for Prospector, a significant result has been the evolution of a methodology to acquire and encode models. This methodology involves such elements as interviewing techniques, principles for determining the overall structure of a model, tools for the interactive construction, modification and testing of models, and the use of performance analysis procedures as a diagnostic tool for evaluating and revising a model. The methodology has been applied in various stages to encode exploration models currently in Prospector with encouraging results, and there is good reason to believe that it can be usefully applied not only to the construction of any new model for the Prospector system, but also to models in other application domains and perhaps to other KB systems as well.

Although the model construction process is not a tightly structured process and is still far from being a routine matter, it does progress through several distinct phases. Using the experience* with the development of mineral exploration models for Prospector, we identify some of the activities that take place in each phase of the model-building process.

2.1. Initial Preparation

2.1.1. The Domain Expert (DE) and the Knowledge Engineer (KE) Learn Each Other's Terminologies

A critical aspect of the KA process is the interdisciplinary communication gap that exists between the KE and the DE. The KE knows generally as little about the domain he is modeling as the DE knows about expert systems or computer science. An important role of the KE (whether a human or a computer program) is therefore to assist the DE in structuring his knowledge so that the vocabulary and concepts he uses to describe the domain and his expertise in solving problems in that domain can be transferred to the expert system.

A common language is established during this initial meeting between the DE and the KE. An hour or two of informal discussion is required, including a demonstration of the Prospector system if this is the DE's first experience with expert systems, giving the DE a good idea of the various ways the system is used. The KE becomes familiar with the general characteristics of the situation to be modeled while the DE becomes familiar with the KE's language for describing models. Typically, the following terms are discussed:

Taxonomies
Rules
Inference network
Semantic-network representation of statements
Spaces

---

* Recent experience with KAS and a description of how the knowledge-engineering process was conducted in actual situations is given in Appendices A and B.

Logical combination of spaces (i.e., AND, OR, and NOT)
Certainties/probabilities, prior probability
Sufficiency/necessity strength of a rule (i.e., LS/LN values)
Spaces requiring numerical input (i.e., L(x) spaces)
Contexts.

2.1.2. Listing Case Studies to be Covered by the Model

The DE will probably have in mind known cases that will serve as examples for the model to be developed. These may be cases that he intends the model to match (some will match perfectly, others may not match as well) and cases that should serve as "near miss" cases, i.e., cases that are partially favorable but lack certain characteristics that are critical to establishing overall favorability of the model. Such a concrete list of known cases facilitates the identification, during the design phase, of critical factors for favorability that hold generally for the listed sites and makes more apparent the intersite differences that must be accounted for in the model. In addition, this list of known cases will be particularly useful to the KE in testing and analyzing model performance during the last phases of development.

2.2. Initial Design of the Model

2.2.1. Development of a Skeletal (Network) Structure of the Model

To illustrate, assume that the model being developed is the XYZ-model corresponding to the fictitious scenario described in Appendix B. In this step, the DE enunciates his knowledge about the situation(s) to be modeled, and organizes it, with the help of the KE, into a hierarchical network structure (of the sort shown in Figure B-1, Appendix B) without worrying about numerical values and how the inference network will be interpreted by the consultation system.

A typical dialogue between the DE and the KE during this step might be as follows:

KE: Can you identify the principal general factors required to establish general favorability?

DE: We must be in a volcanic province,
the intrusive system must be of the right kind,
and we must identify favorable zones of alteration and mineralization.

KE: Zones of alterations? What's that?

DE: ....

KE: OK.
(The KE draws the spaces PVP, FINT, and FZONES, and connects them to XYZ).
Is it reasonable to ask a user directly about any of these factors?

DE: FINT and FZONES are too general; we have to establish those from field evidence. But we could ask about PVP, although if the user cannot answer that there are ways to establish that from field evidence as well. We could, for instance, ask about the presence of a few volcanic rocks that are commonly found in this kind of deposit.

KE: Fine. Let us remember that.
(The KE notes that it is probably a good idea to connect PVP with the taxonomy of volcanic rocks.)
If there is an intrusive, how do you tell it is of the right kind?

DE: It has to be either plugs or dikes.
(The KE draws the spaces DIKES and PLUGS and connects them to FINT)

KE: Does it make sense to ask the user directly about the presence of these dikes and plugs?

DE: Yes, these can be directly observed in the field.

KE: How about the alteration zones, what kind of observations are relevant for this factor?

DE: Well, we will have to find out what kind of host rock is there and see if we can identify any zones of alteration and mineralization. We are mainly interested in a biotized zone or a pyrite zone.
(The KE draws the two spaces for BIOZONE and PYZONE and connects them to FZONES).

KE: ....

DE: ....

The primary objective of this step is to identify relevant factors to establish or rule out the type of deposit being modeled, and to organize these factors by identifying which are direct observations (field observations) and which will require more elaborate investigation before they can be established.

Using a "top-down" approach to interview the DE, the KE first identifies the principal general factors required to establish overall favorability. If it is reasonable to ask a user directly about any of these factors, they are noted to be "askable" concepts and serve directly as evidence spaces. Many of these factors, however, will be general hypotheses. These are noted to be "unaskable" concepts and will serve as hypotheses to be established by lower level evidences. The DE must identify the factors needed to establish the favorability of each of the unaskable hypotheses. This process of progressive elaboration is continued until every factor is related to observable evidence. Each level of the inference network thus obtained represents a more detailed expansion of the level immediately above it.

As he interviews the DE, the KE starts to sketch the inference network, roughly tree-shaped, with the "leaf" (tip) nodes of the tree

representing direct observations (field observations) and the intermediate nodes representing interpretations and conclusions drawn therefrom. Hence, when the model is complete, the DE will have identified the particular observations that will be the foundation of the model (i.e., the questions that are asked of a user during a consultation session), and will have identified the intermediate levels of conclusions and interpretations that will be affected by these factual inputs.

In this step, as well as in all subsequent steps, the KE notes any new concepts introduced by the DE. These may be simple concepts, which will be added to the taxonomies (such as a mineral or a rock type that may never have been mentioned in any of the previous models), but also new phenomena peculiar to this model, which will require the invention of new representation or inference mechanisms. This was the case in Prospector, for instance, when "alteration zones" were introduced for the first time in one of the models. As a result, the language for representing inference networks was extended to include a mechanism to support multiple instantiations during a consultation of portions of the inference network (see Chapter II Subsection 4.3).

2.2.2. Design of the Inference Network

Inference network design is the most creative and intellectually demanding step of the model-encoding process. The DE must now specify how the different factors identified in the previous step are actually combined in each level of the inference network, and must make numerous choices that will affect the flow of control during a consultation. We still are not too concerned with the numerical behavior of the inference network, but the KE must now begin to investigate the DE's intentions and explain the various options and their consequences. Resuming our dialogue between the KE and the DE, the following might be discussed:

KE: Let us examine the three general factors at the top, (PVP, FINT, FZONES).
Do they each affect your belief in the presence of an XYZ-deposit independently from the other two factors?
Are they equally important?
Do they all have to be established or is it enough if only one factor is established?

DE: They all are important and pretty much independent, but the presence of alteration zones is the most critical factor.

KE: OK. We probably will have rules here.
(The KE marks the arcs connecting the three spaces to XYZ, indicating that they are rules).

KE: How about the order in which the three factors are established, do you have any restrictions or preferences?

DE: Yes, the zones should be established last. Also, it does not make much sense to try to establish the existence of an intrusive unless we know we are in a volcanic area.

(The KE indicates that FINT is a context for FZONES and that PVP is a context for FINT).

KE: How about the effect of DIKES and PLUGS on FINT?

DE: Any one of the two factors will be sufficient.
The same goes for BIOZONE and PYZONE for establishing the alteration zones.
(The KE indicates that FINT and FZONES are both disjunctions (OR)).

KE: ...

DE: ...

In this step, among other decisions, the DE and the KE must make the following kinds of choices:

* Type of connection between spaces: As pointed out in Chapter II Section 3, evidence can be combined through inference rules and logical constructs. Disjunctions (OR) are used when any one evidence is sufficient, conjunctions (AND) when all evidences are necessary, and rules when each evidence contributes individually to the favorability of the hypothesis. Unfortunately, sometimes in practice none of these canonical methods is exactly right, and the DE must either settle for an approximation or work with the KE in constructing a satisfactory network structure combination from these primitive elements.

* Question sequencing and control: The hierarchical structure of the inference network implies that questions will be asked in a depth-first ordering (top-to-bottom and left-to-right). In Prospector, this basic control strategy is enhanced by various selection mechanisms and stopping criteria associated with each kind of connection between spaces. Sometimes, however, a particular order is mandatory because a question may make sense only in a particular context, one that must first be established. Sometimes it is merely conventional to ask questions in a particular order. Finally, a particular question about weak inference should sometimes be asked only after an attempt to draw stronger inference was made and produced inconclusive results. To handle these and similar situations, the DE can use the context mechanism (Chapter II Subsection 4.1) to force the control strategy to ask questions in nonstandard order.

2.2.3. Choosing Numeric Values for Parameters

The inference network constructed so far represents the model's overall structure. In principle, we can begin to run and test this model. In fact we do just that in practice to debug the overall behavior of the skeletal model, since KAS will assign reasonable default values for missing parameters in the description of spaces and for any

missing numerical values (e.g., all terminal nodes are assumed to be askable, nonterminal nodes are assumed to be unaskable, and a default prior probability is assigned to a node when needed if one cannot be computed from related nodes). For this presentation however, we assume that this has been done and we must now get the "first round" of numerical values from the DE.

Before a rule can be applied, the DE must specify the two numeric values LS and LN that indicate the rule strengths--the degree to which the evidence is necessary and sufficient in establishing the hypothesis of the rule. For most spaces, another numerical value, the prior probability, will also be required (Chapter II Section 3).

We resume our interview with the DE:

KE: How do each of the three factors PVP, FINT, and FZONES affect the hypothesis XYZ? You said earlier that FZONES was most important. Tell me, for instance, how discouraged you would be if you learned that there are no alteration zones?

DE: Just slightly. Alteration zones are a very encouraging sign, but their absence is not very critical. It is not unusual for these zones to have disappeared during the last two to three million years due to erosion, but they must have been present at some point in the history of the prospect.

KE: How much would you say the odds on XYZ would decrease if there are no alteration zones? 1/10? 1/100? ...

DE: I would say somewhere around 1/10

KE: OK. How about PVP?

DE: That is just the opposite. It does not matter that much if we are in a volcanic province (volcanic provinces are quite common), but if we are not, that is very discouraging.

KE: How discouraging?

DE: Basically rules out our chances for this type of deposit.

KE: How scarce is it to find plugs or dikes in a volcanic province?

DE: Quite scarce, but plugs are about twice as scarce.

KE: ....

DE: ....

A critically important principle in choosing values for the rule strengths is that of independence of the evidence. The objective is to specify the favorability of a single hypothesis space based solely on its evidence spaces, independently of the rest of the model. This simplifies the task of the DE at this point in the development of the model, in that he need take into account only a few factors at a time and avoid the hopelessly complex task of predicting how the dozens of numeric values will interact when the complete model is executed.

Another choice the DE is asked to make at this point is that of specifying how the question about a particular evidence is to be asked during the consultation, and what kind of answer should be expected from the user (i.e., yes/no question, certainty question, or value question). What kind of information the DE is asked to supply will depend upon the type of question specified. For instance, if the user is to be asked about a range for the value of some parameter, such as geological age, concentration, or grain size, then the evidence is the left hand-side of a variable-strength rule and the DE must supply a distribution function defining the rule strengths for all possible intervals as well as a corresponding function for the prior probabilities of the rule's hypothesis. This means also that during the consultation the user will first need to supply an interval (age interval, for instance) and then a certainty indicating his confidence in that interval.

The task of figuring out large quantities of numerical parameters is naturally the most tedious. The DE sometimes feels uneasy and does not really believe that these numbers are useful because he thinks of all the situations and exceptions in his experience where these numbers are actually wrong, and cannot visualize the overall effect on the model. The KE does not believe either that the numbers are correct but has to get each one of them. He skillfully pushes the DE into the smallest corner possible, and painfully but systematically extracts numbers to fill all his empty slots. Because of the size and complexity of the interaction in parts of the model, the numbers often contain inconsistencies that are detected only when the entire model is analyzed. The REsident Network Editor (RENE), described in Chapter IV, contains tools and features (such as the bookkeeping system and the COMPLETE command) to help both the KE in systematically acquiring all the information he must have, as well as the DE in figuring out appropriate structure and numerical values (the TRY command, for instance).

At this point we have an overall structure for the model. Subsequent steps will deal with the installation, testing, and fine tuning of this model.

2.3. Installation of the Model in the Consultation System

2.3.1. Creation of a Computer File Containing the Model Definition

The KE is now ready to transfer the inference network sketches, which until now were drawn on paper, into a computer-readable format. RENE performs this transfer, the purpose of which is solely to get a first version of the model, which will be then used in the refinement steps ahead. The result is the creation of one or more external files containing a description of the inference networks in a symbolic form (see Subsection 3.1). RENE contains a parser and a bookkeeping system that allow the external files to be loaded repeatedly to recreate and modify the inference networks until the KE is satisfied with the first version. Since RENE uses the internal representation of the model to create the external files, they are assumed to be free of syntactic

errors such as misspelling of keywords in the formal language description, and the bookkeeping system ensures that they are free of logical and functional errors such as missing or inconsistent numerical values, unconnected or missconnected spaces, misspelled space names, spaces that are referred to but not defined, and so on.

2.3.2. First Test Runs

The KE can now use some of the facilities for controlled execution and testing supported by RENE to run the first consultation. The main objective in this first run is to test the basic flow of control of the consultation and see if it roughly corresponds to the intentions of the DE. Because of interactions between effects due to numerical parameters and those due to the structures and kind of connections in the inference network, it may not be easy during the model design phase to predict the flow of control in each situation of the consultation. The behavior of the model at this stage typically results in the KE's making simple modifications that affect the flow of control in the inference network, such as

* Adding context arcs to force a particular question sequencing.
* "Blocking" a few hypotheses (using the BLOCK feature described in Chapter II Subsection 4.2) to force the consultation system to ask all the questions relevant to establishing each of these hypotheses before attempting to establish a different hypothesis.
* Modifying some of the default values assigned by RENE, such as a prior probability for a space or the askability status of a space (indicating whether it is askable and how it is to be asked).

2.3.3. Testing the Model on Known Cases

The DE will now gain initial impressions concerning model performance. He observes the model's behavior in a series of consultations involving the known situations recorded during the knowledge-engineering step described in Subsection 2.1.2 above. These consultations will be repeated in several subsequent steps and each time major changes have been made to the model. To eliminate some of the routine tasks involved for the KE and to dispense with the DE's having to be physically present during these consultations, tools are available in the consultation system to automate the mechanics of the consultation session. The following steps are performed for that purpose:

2.3.3.1. Creating a Questionnaire for the Model

Using the internal representation of the model, a questionnaire data sheet containing all the questions that may be asked during a normal consultation is generated.

2.3.3.2. Filling in Questionnaires for the Known Cases

The DE is asked to fill out a questionnaire for each of the known situations; a data file is created corresponding to each completed questionnaire.

2.3.3.3. Running the Test Cases in Batch Mode

Invoking the BATCHMODE facility of Prospector, the KE runs several consultations, using as input the questionnaire data files corresponding to known situations created in the previous step. He can use the various trace and explanation (the TRACE and SUMMARIZE commands) facilities of the consultation system to observe the model's numerical behavior as well as the flow of control. Modifications to the model in this step may involve the DE and include changes of numerical values (such as rule strengths and probabilities) as well as of the network structures (such as adding new rules). The controlled execution features of RENE permit the KE to analyze and debug small sections of the model until they are individually satisfactory. He can then analyze larger sections and continue to modify the model until the DE is satisfied that, by in large, it reflects his intentions.

Subsequent steps will prepare the model so others can find it helpful in situations other than those used up to that point in the testing. Attention will be given to various aspects of a consultation, including consideration of the potential volunteering of information by the user, interaction between this model and existing ones, and the psychology of interaction with the user. Some of these concerns are peculiar to this model while others are related to the general features of the consultation system and are intended primarily to enhance its behavior during interactive consultation.

2.3.4. Completing the Description of the Model

2.3.4.1. Creating the Semantic Representations of the Nodes in the Inference Network and Assimilation of the Model in the KB.

In all the previous steps we were concerned mainly with debugging the general flow of control of the consultation and the overall performance of the inference network. In this step, a more refined semantic representation is assigned to several spaces (representing statements in the KB), and the taxonomies are extended to contain any new domain concepts that were introduced during the interviews with the DE. This allows the semantic network matcher (Chapter V) to relate the statements to each other, thus preparing for the activation of several powerful features of the consultation system. In particular, the system will be able to accept and make use of information volunteered by the user during a consultation session (mixed initiative). Also, the relations that may be discovered by the Matcher between some of the statements could introduce probabilistic constraints for some parameters of the inference network (priors of spaces, likelihood ratios of rules). When this step is completed, a more

thorough debugging of the probabilistic behavior of the inference network is possible and any numerical inconsistencies should have been uncovered.

This step will also uncover the interaction and overlap that may exist with other models (the model is loaded into the system together with one or more previously developed models that will ultimately be included in the same consultation system), and correct any inconsistencies that may be introduced by the new model. Revisions and modifications to these already existing models may be required.

In previous test runs, the consultation system was working only in the backward-chaining mode. Now that the semantic networks are defined and the taxonomies extended, mixed-initiative consultations are possible. Running a few cases where the KE volunteers a few observations at the beginning of the consultation as well as at strategically selected points during the consultation will permit further debugging and refinement of the flow of control as well as of the model's numerical behavior.

2.3.4.2. Completing the Explanation and Rephrasing Texts

At this point, the KE and DE are reasonably satisfied with the behavior of the model and it is prepared for inclusion in the KB (which will contain this and the previously developed models). In Prospector, the DE is asked to supply various pieces of text that will be attached to appropriate spaces and displayed to the user if he uses commands to that effect during the consultation. For instance, the reason (geological background and motivation) for asking a particular question may be explained in a few paragraphs when the user types the command "WHY" when asked that question during a consultation. Other kinds of texts can be included in the model definition; their purpose is mainly to enhance the interaction with the user, announcing to him, for instance, at appropriate places in the consultation the general intent of the following group of questions or, if he requests it, rephrase a particular question explaining in more detail how he should interpret that question and what answer is expected from him.

2.4. Performance and Sensitivity Analysis

An experimental methodology* has been developed within the Prospector project for measuring quantitatively the performance of Prospector along several dimensions. Several tools have been developed to collect experimental data and to analyse results (see Subsection 3.3 below and [28]). We list below dimensions along which experiments have been designed and conducted to improve the performance of models (as opposed to those intended to improve the inference procedures of Prospector). With respect to each of those dimensions, selected sections of a model or the entire model can be analyzed.

---

* A detailed description of the methodology and the experiments can be found in [19,28].

2.4.1. Accuracy and Consistency in Predicting the DE's Judgment

Because Prospector is intended to emulate the DE's reasoning process, experiments are conducted to assess how well Prospector's conclusions compare with the DE's expectations. The DE is asked to fill out questionnaires with the known deposits in mind (identified in Subsection 2.1.2), covering several stages of exploration (amounting to many unknown or uncertain responses in the early stages of exploration whereas much more information is available after a site has been completely explored). For each case, Prospector's conclusions are compared with the DE's own assessment (using the same means for expressing his assessment as Prospector, i.e., a certainty value between -5 and 5), and the results from each site are analyzed to characterize the consistency of the model (and of Prospector) across several sites.

2.4.2. Robustness

A sensitivity analysis of the model is performed to assess the sensitivity of Prospector's conclusions to perturbations or uncertainties in the field observations (user responses) on which the conclusions are based.* This analysis will help to identify critical factors, i.e., questions that could have an important effect on the final outcome, had the user responded differently.

2.4.3. Ability to Discriminate Effectively between Several Hypotheses

By including cases of known deposits that match a model, but less closely than those used in the design phase Subsection 2.3.3.3 above, and "near misses" cases, i.e., prospect sites that are partially favorable, but lack certain critical characteristics, the model can be "calibrated" to discriminate more effectively between closely related hypotheses.

2.5. Revision of the Model

Performance analysis results establish priorities for model revisions in that they identify the particular sections of a model that would most benefit from improvement, both in regard to concurrence with the DE's intentions and model robustness. Early versions of a model generally will not perform as well as the DE intended, and our experience with Prospector has demonstrated that the model design process can be accelerated by using performance analysis as a routine diagnostic tool for "fine tuning" models.

Clearly, many model changes resulting from any of the design phases described above may require that the KE iterate through several of the previous steps until both he and the DE are satisfied that further improvement and revisions are either unnecessary or too costly.

* Other experiments in which the input data are unchanged but where the rule strengths of rules are modified indicate the influence of the model designer's uncertainties on Prospector's performance.

3. Tools for Encoding Models

The Knowledge Acquisition System (KAS) is a collection of special tools developed for the Prospector consultation system as more insight was gained into the methodology of the model-building process described above. Although the primary objective was to eliminate the more tedious and routine tasks of the KB's encoding process, it evolved toward the design of tools that would prove useful in all aspects and phases of the KE process. In addition, because many features supported in KAS are also useful in some aspects of a consultation, their development has contributed to considerable improvements in the consultation system as well. Of the tools directly available to the KE (as opposed to those that may require knowledge about the system, such as debugging tools), the following classes of tools can be distinguished:

- Tools to maintain external descriptions of the KB
- Tools to assist in the model-building task
- Tools to evaluate model performance.

We will be primarily concerned in this work with tools in the second category, we will describe these in detail in Chapters IV and V and briefly describe here a few important tools in the other categories that are mentioned in other parts of this report. Additional details can be found in previously published reports and technical notes of the Prospector project [18, 19, 20, 21, 28].

3.1. Tools to Maintain an External Representation of the Knowledge Base

These include:

* A formal language for representing taxonomies, inference and semantic networks in a symbolic format so that they may be stored on external files. The primitives of the formal language allow the KE to express all the components of the knowledge base including all the constructs used to describe control and inference information in inference networks. Examples of the description of portions of the KB using this formal language can be found in Appendix B.

  The descriptions can be generated automatically at any point during the KE session using RENE and can be stored as text files. These files can be loaded for subsequent editing using RENE, or they can be edited with any conventional text editor.

* A keyword-driven parser* to interpret the external representation and generate the corresponding internal network structures used by Prospector. When the external files are loaded from RENE, the network structures created

---

* The Parser is the PARSEFILE program [19], developed for Prospector by Kurt Konolige and extended by the author to support the automatic bookkeeping features of RENE.

are exactly as specified by the descriptions, and RENE's bookkeeping system will collect the information it needs about inconsistencies or unfinished business as if the networks were being built in main memory using RENE commands.

3.2. Tools to Assist in the Model-Building Task

Two distinct phases are involved in building a knowledge base for an expert system: the knowledge acquisition phase, where information is extracted from domain specialists, and the actual entering of that information into the computer by the KE. Although specific aids can be devised to provide some assistance in each task, in practice the two tasks are not easily distinguishable. As described above, the model-building process goes through several iterations requiring feedback and intensive interactions between the DE and the KE. We have therefore chosen to develop tools in KAS that can be used by either the DE or the KE, ensuring only that the dialogue management components of the system contain appropriate help features to accommodate the needs of a variety of types of users as well as the requirements of many different applications. We hope that at some point knowledge bases can be built entirely by the DE, perhaps after attending a "knowledge-base-building workshop" where he receives guidance as to how to organize and express his knowledge in the rule-based formalism, as well as some basic knowledge about computer operations.

The two main components of the KAS system are:

* A network editor, RENE (REsident Network Editor), with an automatic bookkeeping capability that provides features to assist both the DE during the knowledge acquisition phase and the KE in all other phases of the model building-task. The network editor is described in Chapter IV.

* A semantic network matcher that is being used in Prospector as a knowledge acquisition aid but that also provides assistance to the Prospector consultation system such as dialogue management and probabilistic consistency checking. The Matcher is described in Chapter V.

3.3. Tools for the Evaluation of the Performance of Models

The performance evaluation task described above requires test cases to be evaluated and analyzed by Prospector. Several tools have therefore been developed to mechanize the process as much as possible; they are discussed below.

3.3.1. Questionnaire and Input File Generation

A questionnaire, basically a listing of all the questions that might be asked during a normal consultation, is generated from the internal representation of the model and therefore also contains instructions related to the flow of control and question sequencing.

For instance, the following instruction may precede a question: "Answer this question only if you answered positively the previous question and question 65, otherwise answer question 74."

When the questionnaire is filled out, a corresponding input file is generated that will contain the user's answers to the questions and that can be run through Prospector in "batch mode." Both the questionnaire file and the input file are "incremental" in that they are updated when the model is modified.

3.3.2. Batch Mode/Save Mode

The SAVEMODE feature allows the user's answers to be saved on an external file and the BATCHMODE feature allows Prospector to use an external file as input for a consultation.

Designed initially as a debugging tool to avoid the tedious job of having to repeat a long consultation before a situation could be recreated, these features were easily extended to become a powerful performance evaluation tool, particularly for sensitivity analysis. In addition to providing a convenient noninteractive mode for the KE to run several consultations using external input files, the BATCHMODE feature allows him to specify how he would like the answers in the input file to be varied during the consultation. The options include a systematic increase or decrease of the answers, (thus yielding "more certain" and "less certain" runs), but the KE can also specify any arbitrary procedure (INTERLISP subroutine) for making the changes. The BATCHMODE and SAVEMODE features can be used simultaneously so that the modified answers used in a BATCHMODE consultation can be saved for a subsequent consultation and modified again if desired. The KE can also specify that only the answers to selected questions are to be retrieved from the input files, whereas the other questions as well as those for which no answer was recorded in the input file will be answered from the terminal or the KE can specify a procedure (again any INTERLISP subroutine) for handling these questions.

IV RESIDENT NETWORK EDITOR (RENE)

The formal language and corresponding parser (mentioned in Chapter III Subsection 3.1) are useful tools for maintaining an external description of the KB. The external files containing the symbolic descriptions are ordinary text files that can be created and modified with a text editor. These tools, however, have the usual disadvantages of text editors and compilers, most of which stem from the isolation of the KB description from its representation, and of the editing processes from the execution processes (see for instance the discussion of "residential" versus "source file systems" in [59]). Our work on developing knowledge-engineering tools to assist in tasks such as those described in Chapter III Section 2 was directed at overcoming such problems by integrating the model development process with the representational and computational formalisms of the host consultation system.

In Prospector, various kinds of networks encode the knowledge base (see Chapter II Section 2). Their attraction centers largely around two factors. First, the expressive power of networks is sufficient to encode any fact or concept that is encodable in any other formal, symbolic system such as those based on predicate calculus [68, 70]. This means that networks can serve as a common medium of representation for diverse kinds of knowledge. Second (and this distinguishes network structures from other formally complete systems), network structures that encode information can themselves serve as a guide for information retrieval. From a given node, nodes representing related items are found simply by following pointers from the node to its neighbors. Thus, a network provides its own meaning-bearing indexing system. To the extent that the labels on arcs and nodes are meaningful to network-manipulating procedures, they provide guidance to help traverse the network in search of information relevant to a task. Contrary to conventional databases, networks do not merely furnish a static repository of information that is to be shared by multiple modules; rather, they can serve as a medium of communication between modules. The semantic network representation of statements, for instance (see Chapter II Subsection 2.2), provides the basis for communication between the general-purpose knowledge such as taxonomies and the special-purpose knowledge such as rules. Examples of this role of networks will be illustrated in Chapter V. Further discussion of the representation issues as well as a more detailed description of the full capabilities of semantic networks can be found in [39].

To assist in the creation and maintenance of the wide variety of network structures that constitutes the KB of the Prospector system, a REsident Network Editor (RENE) constitutes the principal knowledge-engineering tool of Prospector. It consists of a collection of specialized tools and features including:

* A structure editor to build and modify networks
* An automatic bookkeeping system
* A dialogue management system
* Facilities for controlled execution of inference networks.

A comprehensive list of the features supported in the current implementation will be found in Primer form in Appendix C and a sample knowledge-engineering session using RENE is presented in Appendix B.

Let us consider each major component of RENE in turn.

1. Structure Editor

The advantages of structure editing as compared with text editing are well known to users of structure editors for programming languages such as INTERLISP [66]. For example, the primitive operations of a text editor used to create an external file containing the description of a Prospector model are string operations such as changing, inserting, or deleting characters. These operations are only indirectly related to the knowledge-engineering operations described in Chapter III. When creating or modifying a model, the KE (at least in a rule-based environment such as that of Prospector) thinks in terms of operations such as connecting or disconnecting sections of an inference network, adding or deleting rules, or creating a semantic encoding of a statement.

As described in Chapter II Section 2, the KB of Prospector contains a variety of networks (inference networks, taxonomies, and semantic networks), each with a different internal representation. Because RENE resides in main memory together with the networks on which it operates, the editor commands will operate directly on the representation of these networks. Thus, in addition to the usual capabilities of editors to examine, create, delete, and modify structures, RENE allows the KE to work in terms of the different kinds of "nodes" and "arcs" without knowing the details of their internal representation or particular implementation.

2. Automatic Bookkeeping System (BS)

In the earliest stages of model-building, only incomplete and dispersed pieces of knowledge are available, and in most instances, neither the DE nor the KE has any clear idea of exactly how the final model will look. Considerable time could therefore elapse between making a mistake in an early development stage and its later detection. Moreover, more than one KE could be involved in developing a model. The chances for making blunders, such as misspelling space names, forgetting needed information, or misconnecting parts of the network, are present in each phase of the model-building process.

To accommodate the highly iterative and incremental nature of knowledge-engineering, an automatic bookkeeping system (BS) is built into RENE, the primary objective of which is to bring enough confidence

and flexibility into the model-building process to permit the DE and the KE to concentrate on the design of the model itself rather than on the process of entering it into the computer. The BS does this by "watching over their shoulders" during all phases of development, protecting against various kinds of common errors (such as accidentally disconnecting sections of a network), assisting the KE in acquiring and entering the necessary information, and keeping records of unfinished work (such as missing parameter values).

The BS uses general knowledge about network building as well as knowledge about many of the special properties of Prospector networks to assist the KE in editing the KB. It uses knowledge about Prospector's computational formalisms to protect against illegal parameter values and automatically assign reasonable default values wherever possible. These bookkeeping facilities permit the KE to develop several sections of one or more models concurrently without worrying about getting confused. He can enter the various elements of the KB (rules, control information, taxonomies, priors, likelihood ratios, and so on) in the order he finds most convenient or that in which he receives the information from the DE. At any time the KE can inquire about unfinished business and have the BS prompt him to supply any or all the missing information.

The operation of the BS is driven by the knowledge it has about several aspects of the model-building process and of the environment in which the model is designed and executed. The nature and organization of that knowledge are critical to the understanding of the bookkeeping tasks supported in RENE. In this section we first characterize the bookkeeping knowledge, give an idea of how it is used in the BS, and describe briefly how the BS operates.

2.1. Hierarchical Structure of Bookkeeping Knowledge

In Chapter II Section 2 we described how general concepts of the domain (such as taxonomies and relations) are represented by network structures in a global database, how collections of nodes and arcs in this global database are grouped together to constitute the (semantic) representation of statements, and finally how statements are grouped together into inference networks. "Zooming" at a node in an inference network will thus reveal its semantic network representation and further investigation of the nodes and arcs in these networks will lead to the global database.

From the user's point of view, many of the editing operations performed on a network structure should be independent of the "interpretation" (see below) associated with a particular network. For example, disconnecting two nodes in an inference network, a taxonomy, or a semantic network, entails the same basic operation of destroying the arc connecting the two nodes. The consequences of the actual operation will depend, of course, upon the kind of network being edited and the representational formalisms associated with the different types of nodes and arcs in that network. This is particularly important during the early stages of development where the networks are incomplete. In an inference network, for instance, the KE may not have made up is mind yet

about the kind of connection needed (e.g., rules versus logical connections), but should be allowed to create or delete the connections.

The knowledge needed for watching over the KB building process is organized in a bookkeeping hierarchy paralleling the design steps outlined in Chapter III Section 2. Each design step results in the extension along some new dimension of the KB's description developed in previous steps. Each level in the bookkeeping hierarchy contains the necessary information for expanding and further specifying the description of the KB along these dimensions. In the earliest stages of model development, for instance, the BS will assist in checking simple "topological" features of the skeletal design, while more sophisticated hand-holding could be required in the latest stages where a modification to the KB may trigger revisions of decisions made in previous steps.

The hierarchical organization of the bookkeeping knowledge in the BS provides a certain level of generality, because each level of description includes information relevant to one or a few aspects of a KB system. In particular, those elements in the description of the KB that are dependent upon its representation, as well as those that depend upon the inference and control mechanisms of the host consultation system, have been identified and isolated. It is thus conceivable that in a different environment, the information related to some aspect of the description may be substituted with one more appropriate without affecting the overall operation of the BS (see Appendix A).

Let us now examine what kind of information is included in the BS.

2.1.1. General Knowledge about Networks

In this level the BS knows only about the two basic elements of networks, NODEs and ARCs.* It knows how nodes and arcs relate to each other to form network structures. In English, the BS's general knowledge about networks can be stated as:

* Each arc has a from-node and a to-node.
* There are two special categories of nodes: the root-node which has no parent node, and leaf-nodes (tip nodes), which have no descendants.
* All nodes except root-nodes must have at least one parent-node, (i.e., there must exist a path from each node to some root-node enabling the BS to keep track of unconnected portions of networks).

---

* General assumptions are made here that networks are used for the representation of the KB. However, nodes and arcs are only symbols to the BS, so the techniques are applicable in any environment. In a different application we could have used the symbols "arrays" and "property lists," or "desks," "chairs," and "classrooms."

2.1.2. Knowledge about the Interpretation of the Networks

An interpretation is associated with a network by defining a mapping between the elements of networks in the previous level (the nodes and arcs), and the concepts they represent. The interpretation reflects how the network is actually used. For instance, the three kinds of network in Prospector's KB (inference networks, semantic networks, taxonomies) are defined by the following mappings:*

```
Inference networks: node = (evidence, hypothesis)
                    arc  = (rule, logical-connection, context-arc)
Semantic networks : node = (entity, place, process, relation)
                    arc  = (arg1, arg2, ... )
Taxonomy          : node = (taxnode, set-of-taxnodes)
                    arc  = (e, s, de, ds, ... )
```

Items enclosed in parentheses () indicate alternatives and the mappings instruct the BS that:

* In an inference network, nodes may be referred to as "evidences" or "hypotheses" (which correspond to from-nodes and to-nodes, respectively), and arcs can be "rules," "logical connections," or "context" arcs.
* In a semantic network, nodes might stand for entities, places, processes, or relations, whereas arcs are merely indices for identifying the arguments of the relations and processes. (The labels "arg1," "arg2," etc., indicate the arguments. Note in particular that in the Prospector implementation a relation is represented as a node rather than an arc to allow for n-ary relations).
* In a taxonomy, nodes stand for the basic concepts of the domain and the arcs indicate the subset/superset relationships between the entries in the taxonomy ("e" for element, "s" for subset, and "d" for distinct).

2.1.3. Knowledge about the Representation of Nodes and Arcs

This level describes a mapping between the elements defined in the previous levels and the data structures for representing them. For instance, an evidence or hypothesis (a "node" in an inference network) is represented by a space because it often contains complex semantic structures, whereas a much simpler record structure represents a node in a taxonomy. The information included in this level enables the BS to automatically select the appropriate representation in different situations. Because the information needed to choose the appropriate representation may not be available in early stages of development, the BS will keep track of the information it receives and select the right representation when the situation is further specified.

---

* The information included in the current implementation of the BS contains additional categories of arcs and nodes. This mapping illustrates, however, the type of information included in this level.

2.1.4. Knowledge about the Inference Mechanism

The bookkeeping information included at this and subsequent levels provides a more specialized characterization of the KB and is highly dependent upon the representation formalisms used in Prospector and the consultation environment itself.

In this level, the BS is instructed about properties of the Bayesian method for computing probabilities:

* Likelihood ratios (LN and LS) are associated with each rule in the inference network.
* A prior probability is associated with nodes in the inference network.

The BS also contains information about the numerical constraints these parameters must satisfy as well as equivalent methods of obtaining them from the user. In particular, many DEs do not feel comfortable expressing rule strength in terms of the likelihood ratios LS and LN. It is often more convenient to express the rule strengths in terms of the direct effect on the consequent of the rule. In Prospector, the DE may specify rule strengths by specifying any of the following pairs of values (see Chapter II Section 3): (LS, LN), (P(H|E), P(H|~E)), or (C(H|E), C(H|~E)). The BS knows that all three options are equivalent and will perform any conversions necessary to communicate with the KE as well as to perform its bookkeeping tasks.

2.1.5. Knowledge about the Consultation System

This level contains information related to dialogue management and flow of control during a consultation. The BS is instructed, for instance, that for each node in the inference network it needs to establish whether the node is askable, how it is to be asked (what text string is typed to the user), what kind of answers are allowed (e.g., yes/no answers, a degree of certainty, specific value or a range of values, and so on).

2.1.6. Knowledge about the Domain

The BS can use taxonomies and models that have been previously developed as well as the taxonomies and the model that are currently being developed as background knowledge to assist the KE. The Matcher described in the next chapter helps integrate the new model in the existing KB and avoids inconsistencies in the inference networks being created. It also furnishes the KE with a convenient way to access nodes by specifying their content and to move around the networks without having to keep track of labels associated with nodes. Commands are available in RENE, which enable the KE to say something like: "Go now to the part of the model where we were talking about chalcopyrite," or "Tell me the names of all spaces that mention pyrite and chalcopyrite."

2.1.7. Defaults

The BS contains information enabling it to automatically assign reasonable defaults in most situations. As a simple example, the BS knows that a terminal node (a leaf node) in the inference network is a space that always represents askable evidence, but that nonterminal nodes represent statements that may or may not be askable. Thus, it will automatically assign a terminal node an "askable" status, but will put the status of a nonterminal node on the list of missing information for that node. Similarly, a logical space will be assigned the default "unaskable" status. If subsequent editing causes a terminal node to become nonterminal (or vice versa), the default value assigned to the askability status is updated. The askability status for nonterminal nodes will be removed from the list of missing information associated with the node only when the value is actually confirmed by the KE.

2.1.8. Error Protection Mechanism

The BS uses the information included in all the levels described above to provide protection against various kinds of errors in all phases of the model-building process. For instance:

* It uses its general knowledge about networks to protect against errors, such as accidentally disconnecting sections of networks. It will also warn about any situation that could lead to ambiguity or confusion, such as duplicate names of spaces or rules within the same model.

* It uses its knowledge about the inference mechanism to protect against illegal numerical values, thus providing argument-type checking and value-range checking capabilities similar to those available in programming languages (such as Pascal) and many compilers. For instance, a prior probability will be rejected by the BS if it is outside of the 0 to 1 range. A legal value of a prior for a space may be subsequently rejected if it is inconsistent with the prior of related statements (see Chapter II Subsection 3.5.2). In the first case, the type was incorrect (not a probability), whereas in the second case the value range was incorrect. Similarly, although both values of the LS and LN for a rule may be independently valid, the pair may be rejected. (If the LS value for a rule is greater than 1, then the LN value cannot exceed 1. See Chapter II Section 3).

* It uses delineations (Chapter II Subsection 2.3) to detect errors in the semantic descriptions of statements in the KB.

2.2. Operation of the Bookkeeping System

The operation of the bookkeeping system is driven by an "add/delete-lists" mechanism similar to that found in many AI systems

(STRIPS, for example [24]). A NEEDS-list can be associated with any element of network in any level of the bookkeeping hierarchy outlined above. Its purpose is to instruct the BS about all information that must be specified when such an element is created. A list of ACTIONS is in turn associated with each NEED in the NEEDS-lists, specifying the consequences of that need's being deleted from (the delete-actions), or added to (the add-actions) the NEEDS-list. Because of the hierarchical organization of the bookkeeping knowledge, the NEEDS-list for an element contained in one level is constructed from the NEEDS associated with that element in that level and from the NEEDS associated with the more general descriptions of that element in higher levels. For example, the NEEDS-list associated with a node in the inference network is

(UPCONNECTION CTYPE ASKABLE PRIOR DESCRIPTION)

The UPCONNECTION need comes from the NEEDS-list associated with a node in the most general level of description, and indicates that (in any kind of network) the node must be connected upward (general knowledge from Subsection 2.1.1).

The CTYPE need comes from the NEEDS-list associated with a node at the "interpretation" level and indicates that the type of connections to the node must be acquired.

The ASKABLE and DESCRIPTION needs come from the NEEDS-list associated with nodes in the inference network, as part of the knowledge the BS has about the consultation system (from Subsection 2.1.5), requiring the KE to specify the askability status of a node and a text string description to be used during the consultation.

The PRIOR need comes from the NEEDS-list associated with nodes in the inference network (evidences or hypotheses), as part of the knowledge the BS has about the inference mechanism (from Subsection 2.1.4) requiring that a prior probability be assigned each node in the inference network.

When the node participates in an editing operation, the NEEDS-list is updated and the ACTIONS associated with any affected NEED are triggered. For instance, to connect two evidences E1 and E2 to the hypothesis H with logical connections of type "AND," the KE can use the following RENE-command:

```
** CONNECT AND FROM E1 E2 TO H  .
```

The UPCONNECTION need will thus be deleted from the NEEDS-lists of E1 and E2 and the CTYPE need from the NEEDS-list of H. A simple example of delete-ACTIONS associated with the two needs UPCONNECTION and CTYPE are these that trigger the automatic computation or updating of another NEED such as the PRIOR associated with each node in the inference network. The actual expressions interpreted by the BS might correspond to

(TO-NODE: (COMPUTENEEDS 'PRIOR) ...)

for the UPCONNECTION need, and to

(COMPUTENEEDS 'PRIOR)

for the CTYPE need.

They both indicate to the BS that it may now be able to compute the prior probability of H (the to-node of the connection). Because H is a conjunction, the prior can be deduced from that of E1 and E2 (if they have a prior). If the BS is able to compute the prior of H, then PRIOR can be removed from the NEEDS-list associated with H, which, in turn, triggers the delete-ACTION associated with PRIOR (resulting in propagating the computed prior through the inference network and checking for contradictions).

If the KE later deletes any of the connections from E1 and E2 to H (or changes the type of the connections so they become rules, for instance), the BS will update the NEEDS-lists of the nodes and arcs involved and the add-ACTIONS that are triggered will reassess the previous consequences and reflect the new situation.

2.3. Extending the Capabilities of the Bookkeeping System

The add/delete-ACTIONS furnish a well-structured scheme for extending the capabilities of the BS. However, most ACTIONS are implemented as procedures tailored to handle particular situations in Prospector so different procedures may be required in a different environment.

> The NEEDS-lists, on the other hand, represent purely declarative information (each NEEDS-list is the result of "compiling" the NEEDS-lists associated with more general levels of description), and can be easily extended.

The KE can associate additional needs (or delete some) with any generic term used in the description of the various types of nodes and arcs occurring in his networks. (This is accomplished with the RENE commands ADDNEED/DELNEED described in Appendix C). For example, the KE can declare a new NEED for rules, say a REFERENCE to the expert who provided it, by adding REFERENCE to the NEEDS-lists for "<rule>." The BS will then expect the KE to supply that need for each rule created. The KE will be prompted for a REFERENCE when he decides to complete the description of any rule that does not contain it.

The BS maintains a representation of the hierarchy of all types of nodes and arcs currently used in the KB together with the NEEDS-lists associated with them, as a mechanism for the inheritance of bookkeeping information. The KE can declare a new kind of "node" and associate a different set of needs with it. For example, the NEEDS-list of a "root-node" of a network contains only one need, the CTYPE. In an inference network, the NEEDS-list of a "top-hypothesis" will also contain CTYPE because it is a "root-node," but also two additional needs associated with nodes in inference networks: DESCRIPTION and PRIOR. Other nodes are recognized by the BS as special kinds of root-nodes and will therefore inherit the NEEDS-list of a root-node augmented with any

particular needs associated with the type of node (e.g., the UPPCONNECTION need is associated with all types of nodes and ASKABLE is associated with nodes in an inference network).

3. Dialogue Management System (DMS)

A crucial issue in the overall design of any knowledge acquisition system is how the man-machine interaction is supported. Several experiments that highlight many issues of concern can be found in [37]. In [77] the two main interaction methods, the question and answer dialogue and the command dialogue, are explored and their properties thoroughly compared. None of these methods alone, however, provides a satisfactory solution to the interaction requirements in KAS:

* To provide the KE with some flexibility in designing the knowledge base, and because the DE does not usually make available to the KE all the needed information at once, a knowledge acquisition system must leave the initiative of conducting the KE session in the hands of the KE. For this purpose, KAS must adopt the command dialogue as the primary method of interaction.

* Unfortunately, with this convenient environment for entering the knowledge base, we immediately introduce new difficulties for the KE, who now needs to keep track of what has been entered so far and what is still missing in each stage of development. The complexity of the model-building task, which involves the design of a model that must not only capture the DE's intent but adapt as well to a particular consultation environment, would suggest that the system should be given more control in monitoring the KE process.

* In addition, RENE is intended to be used by a wide spectrum of user categories, differing either in their backgrounds or in how they use the system. It can be used by the casual user during a consultation to perform a simple alteration such as a change of an answer or of a rule's strength, or it may be used by the experienced KE to design and test an entire model. It is also intended to be helpful at any stage of model development and several DEs and KEs can be involved in various steps of the model design. Thus, some steps could conceivably be performed by "secretaries" having only rudimentary familiarity with the system, while other steps could require a computer scientist, a KE with understanding of the Prospector system, or a KE with knowledge about the domain of application. Because most users are unlikely to spend much effort to master a sophisticated command language, the "question and answer dialogue" method or a "menu-driven" dialogue would thus seem more appropriate.

In developing the Prospector consultation system, similar issues were considered. The most important aspect of these considerations, however, centered around the psychology of interaction with a variety of users and in a variety of situations that can occur during a consultation. As the result of several experiments, the interaction methodology that was finally adopted supported a "mixed-initiative" dialogue (where the system asks questions, but the user is able to affect the course of the consultation by volunteering observations or using special commands to direct the system's attention on a selected hypothesis), combined with a "natural language" approach in communicating with the user. This combination has been found necessary and appropriate in the KAS environment as well. We will describe below each component of this combination:

3.1. "Mixed-Initiative" Component of the Dialogue Environment

The KE communicates with KAS primarily through a command language. However, by embedding a variety of specialized tools in RENE, such as the bookkeeping system (described above) and the Matcher (described in the next chapter), RENE effectively relieves the KE of most concerns with the consultation system in which the model is to be embedded. The KE may voluntarily relinquish control by asking the BS to enter a prompting mode and assist in filling in missing information. In addition, the system will automatically take control any time an inconsistency is encountered to guide the KE in recovering.

3.2. "Natural Language" Component of the Command Language

Once the command language interaction method has been chosen as the primary vehicle for conducting the dialogue, much effort must be spent in the design of the command language. An important consideration in the KAS environment is that, because of the wide spectrum of user categories, it should be easy to use. To this end, RENE contains on-line help features to assist the KE in using the commands.

The Command Interpreter is a parser driven by a grammar that defines the syntax for the arguments of each command. The grammar can naturally be used to check the legal syntax of a command. It has been our experience, however, that even the most developed grammar and the most sophisticated parser are nothing more that a restriction of the valid sentences of the command language. Commands will still be rejected by the parser and both inexperienced and experienced users will be frustrated when their commands are misinterpreted. In RENE, the grammar is primarily used by the novice KE to interrogate the system about the proper use of each command and he can explore the grammar to any depth. The grammar can also be used by RENE to prompt the KE for unspecified arguments. The advantage of this approach is that the grammar can be very simple and still be useful in assisting most user categories. The LIFER system [38], an application-oriented system for creating natural language interfaces between existing computer programs and users, was first used in RENE for our experimentation with this approach, but has been later replaced by a much simpler parser that

interprets the syntax of the commands for the sole purpose of prompting the KE for the arguments.

How well the command language will adapt to different user profiles or different applications will naturally depend upon the richness of the grammar and the capabilities of the parser. The prompting feature is therefore interfaced with the bookkeeping system to support type and value checking of the arguments as they are supplied by the KE. Furthermore, because the knowledge of whether or not an item belongs to a given set is essential in defining grammars, the taxonomies are easily interfaced with the grammar that defines the command language. The richness of the vocabulary of the command language is thus automatically increased as the knowledge base grows.

4. Facilities for Controlled Execution

The knowledge-engineering process (as described in Chapter III Section 2), is a highly iterative process where the basic mode of operation corresponds to a "modify and test" refinement cycle. It is also an incremental process because the DE must be allowed as much flexibility as he needs in supplying the knowledge, and the KE in encoding and testing it. They must have the flexibility to work on any section of the model in any order, as well as the flexibility to work on any aspect of the design in any order.* Although in the outline of our methodology, we favored working on the flow of control before attacking the numerical behavior aspect of the design, if the DE happens to have a couple of numbers on hand while the first sketches of the inference network are being drawn, it can only be helpful to include these numbers in the current design.

In most situations, the KE will therefore need to have immediate feedback on the consequences of changes to the KB. He needs to examine not only the consequences on the network structures themselves (such as knowing that the right arc has been disconnected), but more importantly the consequences on the performance of the model during a consultation. Adding a rule or modifying a rule strength are simple editing operations, but the effect of their propagation through a large inference network may not be easy to predict.

RENE can be used at virtually any time in the development cycle. For instance, after any modification to the KB, the KE may invoke the consultation system specifying that it evaluates a selected section of the model. He may then volunteer observations exactly as a user would do in a regular consultation, answer questions asked by the system and examine the consequences using the explanation system of Prospector.

---

* This is helpful not only because all the information may not be available at once, but also (and more importantly) because RENE is intended to assist in acquiring some of that information. In fact, commands are available in RENE to help the DE figure out numerical values (such as prior probabilities and likelihood ratios) to achieve a particular effect in the inference network (see the TRY command in the primer (Appendix C), for instance).

Each section may be debugged independently from the rest of the model and the modifications retained if satisfactory. Similarly, RENE can be invoked in the middle of a consultation, and execution may be halted to modify the KB, change answers to previous questions, perhaps volunteer some observations, perform a few controlled experiments and then resume the consultation. In both cases, intermediate results, when satisfactory, can be saved in external files.

By combining the full capabilities of the Prospector Consultation system with those of a sophisticated structure editor, RENE provides a convenient environment in which to develop and test models, very similar to that provided for, for instance, by INTERLISP to develop and debug large computer programs [66].

5. Extending the Capabilities of RENE

We have explained above how the capabilities of the bookkeeping system can be extended. In addition, the command repertoire of RENE can be extended by using the ADDCMND command described in Appendix C. The command interpreter is driven by a set of properties associated with each command, and the KE can add new commands or modify any exsting command simply by issuing declarations that will affect the desired properties. Some of those properties govern the interaction between the KE and the system (e.g., the syntax and type of the arguments, how the answer is to be displayed to the user) while other properties will affect the execution of the command (e.g., the actual procedure associated with the command or class of commands).

V SEMANTIC NETWORK MATCHER

Partial-matching (also referred to as interference-matching or correspondence-mapping) touches several issues of representation and efficiency in many AI systems [34]. Its role in Prospector has been significant because it supports many features that make both the consultation system and the knowledge acquisition system (KAS) behave intelligently. One of the most encouraging facts noted in this work is that a matcher need not be very sophisticated to be useful. In fact, we will show that most features described here rely only on the capability of the Matcher to detect simple relationships between statements in the knowledge base and that these features can be supported independently of how the statements are represented. In particular, given any two statements S1 and S2, we will illustrate applications where the Matcher needs only to determine which of the following situations applies:

S1 and S2 are identical (S1 = S2)
S1 is a restriction of S2 (S1 $\subset$ S2)
(or S2 is a generalization of S1 (S2 $\supset$ S1))
S1 and S2 are disjoint statements (S1 $\cap$ S2 = ϕ)
S1 overlaps S2 (otherwise)

After an overview of how the Matcher operates in the Prospector environment, we describe features that are useful during the knowledge acquisition phase (involving the building and testing of the knowledge base) as well as features that are of assistance during a consultation.

1. Operation of the Matcher in the Prospector Environment

1.1. Examples

To illustrate a few common cases, suppose the KB contains the following statements:

S1: "rhyolite is present"
S2: "a rhyolite plug is present"
S3: "an igneous intrusive is present"
S4: "rhyolite or dacite is present"
S5: "pyrite is present"

As these statements are being entered into the KB in the above order, the Matcher will make the following conclusions:

* S2 is a restriction of S1 (Figure 7) because S2 contains an additional constraint, the form attribute, not occurring in S1.
* S2 is a restriction of S3 (Figure 8) because the corresponding external references are restrictions of each others. The taxonomy of rocks shows that "rhyolite" is a special kind of "igneous rock" and the taxonomy of forms shows that "plug" is a special kind of intrusive form. The Matcher finds the path between the corresponding external references that participate in the same relations ("igneous" and "rhyolite" participate in the COMP-OF relation, "intrusive" and "plug" participate in the FORM-OF relation).
* S1 is a restriction of S4 (Figure 9) because "rhyolite," the external reference of S1, is an element of the disjunction "rhyolite or dacite" and participates in the same relation (COMP-OF) in S4.
* S2 is a restriction of S4 (transitivity, Figures 7 and 9)
* S1 and S3 overlap (Figure 10). S3 has an additional constraint describing the physical entity E3, the additional attribute participating in the FORM-OF relation. On the other hand, the external reference "rhyolite," which participates in the COMP-OF relation in S1, is a restriction of the corresponding external reference in S3, "igneous rocks" (deduced from the taxonomy), and thus, S1 is more constrained than S3 with respect to the COMP-OF attribute.
* S3 and S4 overlap. As in the previous example, because the disjunction in S4 is a restriction of "igneous rocks" in S3, which is in turn more constrained than S4 due to the additional FORM-OF attribute.
* S5 is disjoint from S1 ,S2, S3 and S4 (Figure 11) because none of the external references in S1, S2, S3 or S4 is compatible with the external reference "pyrite" in S5. Pyrite is a mineral, while all other statements involve rock types or form values. There is, therefore, no chain of subset/element arcs that would make "pyrite" compatible with any of the other external references.

Figure 12 shows how all five statements used in the examples above are related to each other.

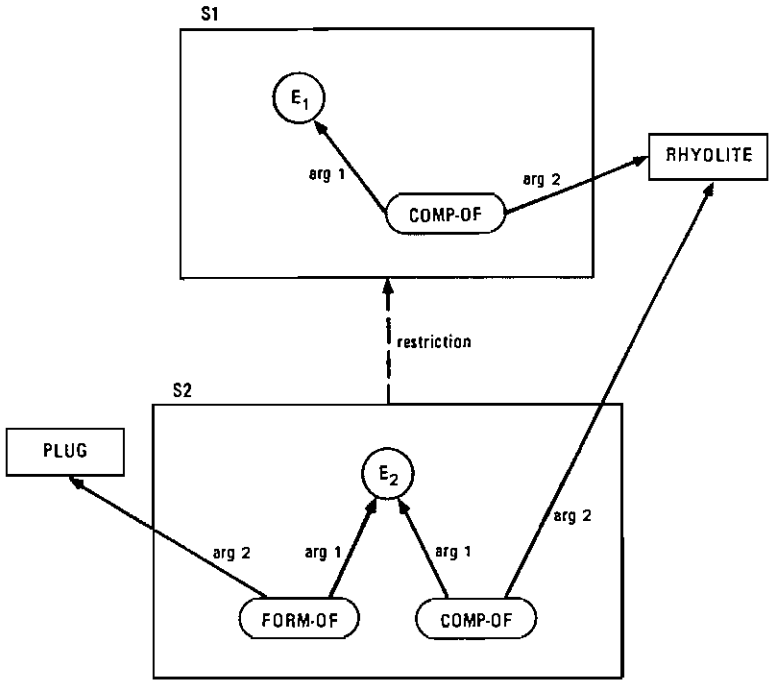


Figure 7 S2 Is a Restriction of S1

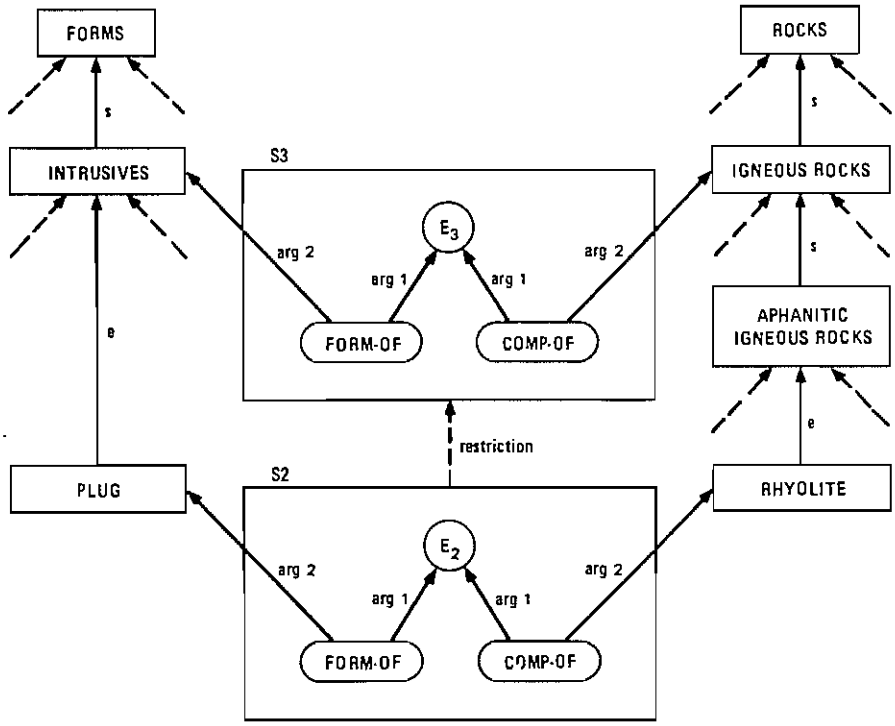


Figure 8 S2 Is a Restriction of S3

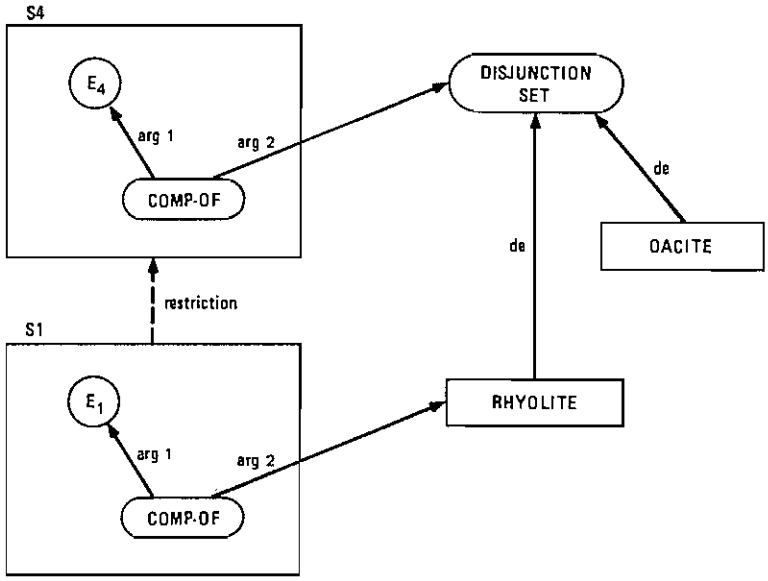


Figure 9 S1 Is a Restriction of S4

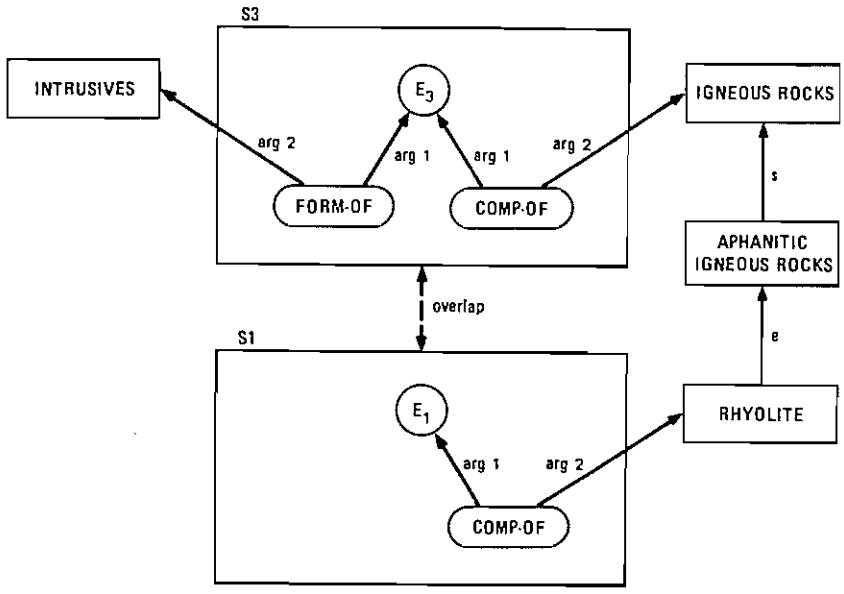


Figure 10 S1 and S3 Overlap

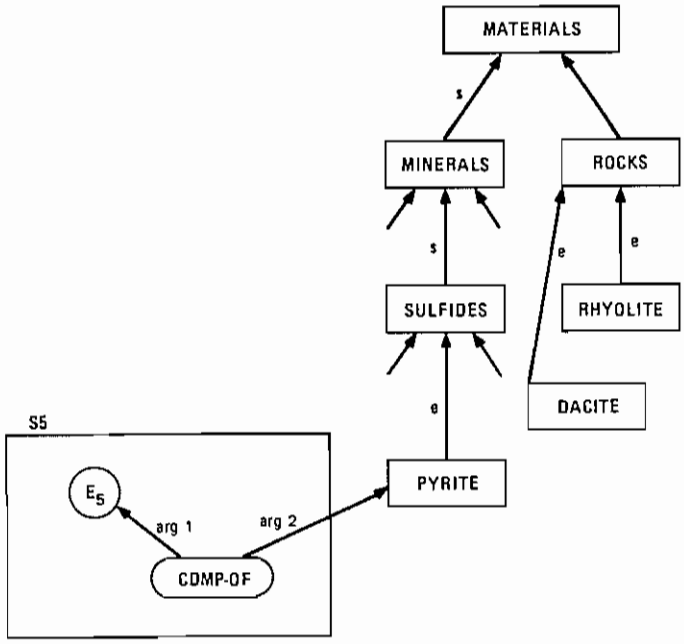


Figure 11 S5 Is Disjoint From S1, S2, S3, and S4

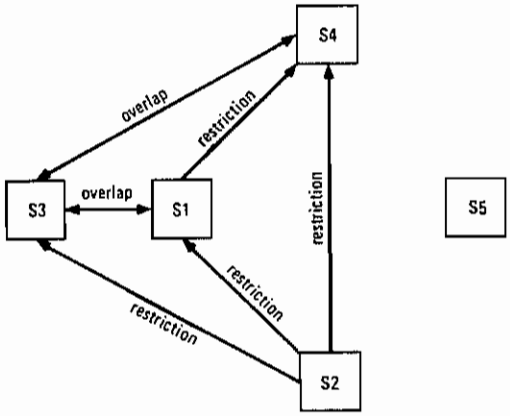


Figure 12 Matcher Links For All Five Statements
S1, S2, S3, S4, and S5

1.2. The Matching Procedure

Let S1 and S2 be the two statements in the KB to be analyzed by the Matcher. We will use indistinctively the same labels S1 and S2 to refer to the statements themselves as well as to their corresponding semantic representations. Let M and N be the number of assertions a_i that

constitute the the semantic representation of the two statements S1 and S2, respectively.* (For instance, for the two statements represented in Figures 2 and 3, the number of assertions are 3 and 5, respectively). Thus,

$S1 = \{a_i\}_{i=1}^{M}$ and $S2 = \{a_i\}_{i=M+1}^{M+N}$. Let

$P1 = \{p_i\}_{i=1}^{m}$ and $P2 = \{p_i\}_{i=m+1}^{m+n}$

be the sets of physical entities, places and processes that participate in the semantic representation of S1 and S2, respectively.

We can group the sets of assertions in S1 and S2 into strict disjoint subsets of assertions such that

$$S1 = \{\{a_{ij}\}_{j=1}^{k_i}\}_{i=1}^{m} \qquad S2 = \{\{a_{ij}\}_{j=1}^{k_i}\}_{i=m+1}^{m+n}$$

where k_i is the number of assertions describing p_i. Thus,

$$S1 = \{A_i\}_{i=1}^{m} \qquad S2 = \{A_i\}_{i=m+1}^{m+n}$$

where A_i is the set of all assertions involving p_i. The grouping must be such that all assertions must be included in some subset A_i, but only once. If an assertion involves more than one entity, it can be associated with any (but only one) of the entities. In particular, the following conditions must be satisfied:

$$\sum_{i=1}^{m,\ m\leq M} k_i = M \qquad \text{and} \qquad \sum_{i=m+1}^{m+n,\ n\leq N} k_i = N.$$

For example, if S1 is the statement "partially biotized hornblende," its semantic representation (Figure 3) corresponds to the seven assertions listed in Chapter II Subsection 2.2:

$$S1 = \{a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\}$$

The seven assertions describe two entities E1, E2, and the process P1. Thus, M=7, m=3. A regrouping of the seven assertions around E1, E2, and P1 (with k_1=2, k_2=2, k_3=3) yields:

$$S1 = \{A_1\ A_2\ A_3\}$$

where

$A_1 = \{a_{11}\ a_{12}\} = \{a_1\ a_2\}$ (Assertions about E1)
$A_2 = \{a_{21}\ a_{22}\} = \{a_3\ a_4\}$ (Assertions about E2)
$A_3 = \{a_{31}\ a_{32}\ a_{33}\} = \{a_5\ a_6\ a_7\}$ (Assertions about P1)

The basic operation of the matching procedure currently implemented in Prospector corresponds essentially to a unification algorithm. In

---

* See Chapter II Subsection 2.2 for definitions.

the procedure described below, each assertion participating in the semantic representations of the two statements S1 and S2 is viewed as a constraint and the Matcher attempts to identify in both spaces, corresponding assertions that participate in the same relation and involve compatible external references. If the external references are not identical, but instead one reference is a restriction of the other, then the assertion corresponding to the more restricted reference is appropriately marked to reflect the additional constraint.

Algorithm

Initially all assertions in S1 and S2 are "unmarked" and all p_i are "unmatched" and "unbound."

For each set A_i of assertions about p_i in S1, do:

Step 1: Choose an unmarked assertion A, involving some external reference E_A and let R be the relation in which p_i and E_A participate.

If there is no such assertion left, go to Step 7.

Step 2: If no binding has been assigned to p_i yet, then find in S2 a corresponding unmarked assertion B, in some A_j containing assertions about p_j and involving the same relation R and an external reference E_B compatible with E_A.

If p_i has a binding p_j, then find an unmarked assertion B in the set A_j of assertions about p_j in S2, involving the same relation R and an external reference E_B compatible with E_A. (p_j is unmatched yet, otherwise it would have been removed in Step 7).

Step 3: If an assertion B is found in Step 2, record that "A matches B" and "bind p_i to p_j" (so that in Step 2, subsequent assertions in S2 will be taken from A_j about p_j until all assertions in A_i about p_i are exhausted in Step 1).

If no suitable assertion B is found, mark "+" the assertion A in A_i and go to Step 1.

Step 4: If E_A and E_B are identical, remove both assertions A and B from A_i and A_j, respectively. Go to Step 1.

Step 5: If E_A is a restriction of E_B, (directly or indirectly deduced from the taxonomy), remove the assertion B from A_j and mark "R" the assertion A in A_i. Go to Step 1.

Step 6: If E_B is a restriction of E_A, remove the assertion A from A_i and mark "R" the assertion B in A_j. Go to Step 1.

Step 7: Record that "p_i matches p_j." Remove from A_i the assertion about the existence of p_i and the corresponding assertion about the existence of p_j from A_j. Mark "+" all remaining unmarked assertions in A_i.

After all A_i have been considered, (whereby Steps 1-7 have been executed m times), one of the following conclusions is reached by the Matcher:

1) If both S1 and S2 are empty, then "S1 and S2 are identical."
2) If only S1 is empty, then "S2 is a restriction of S1."
3) If only S2 is empty, then "S1 is a restriction of S2."
4) If neither S1 nor S2 is empty, then
   a) if any assertion in either S1 or S2 has been marked "R" (that is, if there were some compatible external references participating in their semantic representation), then "S1 overlaps S2"
   b) otherwise, "S1 and S2 are disjoint."

Comments:

(a) As illustrated in the examples below, a statement S2 is a restriction of S1 (or S1 a generalization of S2) when S2 has all constraints that apply to S1, plus some additional constraints. These constraints can be expressed directly through the elements of the semantic representation (additional external references as in Figure 7), or indirectly through the taxonomy (references in S2 are restrictions of those in S1 as in Figure 8).

(b) Clearly, the procedure above does much more than just determine simple restriction/generalization relationships between two statements. Note in particular that we have not fully exploited the implications of the various tags ("+" and "R") associated with each assertion during the matching process. As a side effect in the matching procedure, corresponding (matching) elements in the two statements are identified and information about the quality of their match is recorded. Similarly, the remaining unmatched assertions in S1 and S2 and their associated tags, ("+" or "R"), disclose useful information about the "differences" between the two semantic representations and provide a good basis for computing a "measure" of goodness of the match. For instance, the number of remaining assertions marked "+" in both S1 and S2 is a measure of the differences between the two statements, whereas the number of those marked "R" is a measure of the amount of overlap between them. A further qualification of how close this overlap is can be estimated at the time the external references E_A and E_B are paired. For instance, the number of element/subset arcs separating the two taxonomy entries E_A and E_B could be used as a measure of semantic difference between the two assertions containing them. This measure can be assigned to the individual assertions (at the same time they are being marked "R," for instance) or contribute to some global measure associated with the entire statement. In Subsection 2.2.2.1.2 we show where this additional information can be useful.

(c) Finding a suitable assertion B involving an external reference compatible with that involved in assertion A is straightforward in the Prospector environment because of the semantic network representation of the KB. To find the assertion B, Prospector traces through the taxonomy starting from the external reference E_A of A, until a compatible

external reference E_B is found that is referred to from S2 and participates in the same relation R as E_A in S1.

(d) In Step 7, the Matcher confirms the "binding" of the entity p_i to p_j by recording the match. If for any reason the matching of any of the assertions about p_i and p_j was unsuccessful, this binding is rejected together with all the intermediary matches and markings of corresponding assertions. Using usual backtracking techniques, another suitable binding for the assertion A from A_j, or if necessary from a different A_j, will be fetched in Step 2.

(e) Transitivity of restrictions: The Matcher does not always need to analyze the semantic representations to determine how two statements are related. In many applications the results of previous matches are recorded in some fashion in the knowledge base and the Matcher takes advantage of any such information when available. Particularly, if S1 has been found to be a restriction of S2, and S2 is a restriction of S3, the Matcher deduces automatically that S1 is a restriction of S3.

2. Use of the Matcher

Let us now examine how a Matcher with capabilities similar to those exemplified above can be useful first during the knowledge acquisition phase and then during the consultation.

2.1. Use of the Matcher In Knowledge Acquisition

2.1.1. As a Tool for Knowledge Integration

Building the knowledge base of an expert system is an incremental process. Each time a new statement is added to the knowledge base, the Matcher can determine whether and how that statement is related to any of the statements already entered. There are several ways in which this information is helpful in assimilating the new knowledge.

2.1.1.1. Aid in Maintaining Probabilistic Consistency of the Inference Network

The knowledge bases of many expert systems are organized as explicit or implicit networks of statements connected by rules or logical constructs. Because such inference networks provide the framework for judgmental reasoning, various numerical values, such as rule strengths, probabilities or certainties, are often maintained in them. A major concern of expert systems is the difficulty of keeping the knowledge base error free and consistent in form and content as it grows.

To illustrate how the Matcher assists Prospector in maintaining the probabilistic consistency of the KB, let us consider the simple case where H1 is the most recently entered statement, and H2 is a statement that already exists in the knowledge base such that H2 is a restriction of H1:

Because H2 is a restriction of H1, the set of situations in which H1 is true includes the set of situations in which H2 is true and the probability of H2 can never exceed that of H1. In particular, if the prior probabilities P(H1) and P(H2) do not satisfy this constraint, the violation is detected and a correction will be required. Thus, before a consultation begins, we can assume that $P(H2) \leq P(H1)$.

Unfortunately, even though all the probabilistic constraints are initially satisfied, the probability changes that follow from the use of inference rules during a consultation may not maintain them. For example, if H1 and H2 are the hypotheses (right-hand sides) of two rules E1 --> H1 and E2 --> H2 , and if the evidence (left-hand side) E1 is sufficiently unfavorable for H1, we may have $P(H1|E1) < P(H2)$.* Similarly, if the evidence E2 is sufficiently favorable for H2, we may have $P(H1) < P(H2|E2)$.

The problem is that when the DE provided the rule saying that the evidence E1 is unfavorable for H1, he overlooked the fact that E1 is also unfavorable for H2, and so did not supply a rule of the form E1 --> H2. Similarly, when he supplied the rule saying that the evidence E2 is favorable for H2, he overlooked the fact that E2 is also favorable for H1, and so did not supply a rule of the form E2 --> H1. Indeed, the DE should not be asked to supply rules for such obvious deductions.

The Matcher only helps to detect these situations. It is the responsibility of the KE and of the consultation system to take the appropriate actions to ensure that the probabilistic consistency of the inference networks is maintained during the knowledge acquisition phase as well as during a consultation. As an example, we illustrate actions taken in Prospector for statements that are involved in rules or in logical constructs.**

2.1.1.1.1. Rules

The basic approach is very simple: Let LS_2 and LN_2 be the likelihood ratios*** for the rule E2 --> H2, and suppose that LS_2 is so large that $O(H2|E2) = LS_2 * O(H2) > O(H1)$.

---

* $P(H|E)$ denotes the posterior probability of H given E.

** During the matching process, it is irrelevant how a statement is connected in the inference network. It may appear as a component in a logical construct or as the antecedent or consequent of a rule. However, after the Matcher establishes what kind of relation exists between the most recently added statement and those already in the knowledge base, the role played in the inference network by the statements involved is important in deciding how that information is to be used. The consequences will depend upon the inference procedures used in the consultation system. (See Chapter II for a description of the inference mechanism used in Prospector.

*** Two numbers defining the rule strengths. See Chapter II for definitions.

If E2 is observed, this rule can cause a violation of the probability constraint between H2 and H1 (H2 is a restriction of H1). The balance can be restored by adding a rule from E2 to H1 with a likelihood ratio

$$L\$ = LS_2 \uparrow \frac{O(H2)}{O(H1)}$$

since this rule will change the odds on H1 to

$$O(H1|E2) = L\$ * O(H1) = LS_2 * O(H2) = O(H2|E2).$$

The complete procedure now used in Prospector must take into account various situations that can occur during the consultation and is basically as follows:

Let H1 denote any fixed hypothesis, and let H2 denote any hypothesis that is a restriction of H1. E1 and E2 denote left-hand sides of rules having the right-hand sides H1 and H2, respectively.

Step 1: For every E2: (there may be several rules having H2 as hypothesis)

$$\text{Let } L\$ = \frac{O(H2)}{O(H1)} \max\{LS_2, LN_2\}.$$

If $L\$ > 1$, then

(a) If there is no rule from E2 to H1, create a new rule with
LS = L$ and LN = 1 if $LS_2 > 1$,
or with LS = 1 and LN = L$ otherwise.

(b) If there is a rule from E2 to H1 with likelihood ratios (LS, LN) and $L\$ > \max\{LS, LN\}$,
reset LS to L$ if $LS_2 > 1$, or
reset LN to L$ otherwise.

Step 2: For every E1: (there may be several rules having H1 as hypothesis)

$$\text{Let } L\$ = \frac{O(H1)}{O(H2)} \min\{LS_1, LN_1\}.$$

If $L\$ < 1$, then

(a) If there is no rule from E1 to H2, create a new rule with
LS = L$ and LN = 1 if $LS_1 > 1$,
or with LS = 1 and LN = L$ otherwise.

(b) If there is a rule from E1 to H2 with likelihood ratios (LS, LN) and L$ > min {LS, LN},
reset LS to L$ if $LS_1 < 1$ or
reset LN to L$ otherwise.

This procedure is applied to every pair of hypotheses H1 and H2 that are found by the Matcher to be linked by a restriction/generalization link and may result in the modification of old rules and the creation of new ones ensuring that any probability changes that follow from the use of any single rule will be propagated properly through the inference network without violating the probability constraints corresponding to known restriction/generalization relations. In theory, it is still possible for certain combinations of rules to violate the constraints. For example, suppose that two rules both bear positively on H2, but that neither one separately can raise the probability of H2 above that of H1. In that case, no new rules will be created by the above procedure. However, if both pieces of evidences together can raise the probability of H2 above that of H1, the constraints will be violated. Although the above procedure could be extended to consider the effects of combinations of rules, occurrences of such violations are not common in practice, and the adopted approach seems quite satisfactory. (On rare occasions, in order to avoid such violations, the KE may have to restructure portions of the inference network and "fine tune" it by using an appropriate combination of rules and logical constructs).

In situations where no link has been found by the Matcher between the two hypotheses and in the case where the two hypotheses are linked by an overlap relation, no particular action is taken in Prospector. Although the Matcher often expends much effort to find overlap links, the meaning of such relations is unclear and not much can be said about the probabilistic constraints involved unless additional semantic information is available, (either explicitly in the taxonomies or through some chain of deductions) to help analyze the differences between the two statements that have been identified by the Matcher. For instance, if two statements S1 and S2 are found to be disjoint because the external taxonomy entries in their semantic representations were incompatible, it may be useful to learn if the two entries were mutually exclusive so that if S1 is established during the consultation, ~S2 may be triggered and propagated through the inference network. Similarly, in the case of an overlap relation, it would be useful to learn if the additional constraints in the two statements responsible for the overlap classification are semantically independent or mutually exclusive. After such analysis, an "overlap" relation may be converted to a more useful "restriction/generalization" relation. Although in many cases procedures similar to that outlined above for the restriction/generalization relation could be devised to control probability constraints, overlap relations are used only marginally in other features of the consultation system (see Subsections 2.2.2.1.2 and 2.2.2.2).

2.1.1.1.2. Logical Constructs

If H1 is a logical statement, there are probabilistic constraints that must also be satisfied between H1 and the evidences of which it is composed. The semantic structure of H1 is implied by that of its components and the type of logical connection. The Matcher actually needs never to be called to find the relation between H1 and any of its logical components because the logical connections may be viewed as precomputed matcher-links. A procedure is associated with each type of logical connection (AND, OR, NOT) for computing the probability of H1 given that of its components. In Prospector, for instance, if $\{P_i\}$ is the set of probabilities of the evidences $\{E_i\}$ bearing on the logical statement H1, the probability P(H1) is computed to be:

$P(H1) = \min \{P_i\}$ if H1 is a conjunction (AND)
$P(H1) = \max \{P_i\}$ if H1 is a disjunction (OR)
and $P(H1) = 1 - P(\bar{E}1)$ if H1 is a negation (NOT).

2.1.1.2. Merging of Several Knowledge Sources

Models exist in Prospector as external files and a consultation system is created by "loading" some or all of the models needed for the particular application or user. Because in most applications models will describe portions of the knowledge base relevant to the same domain, a certain amount of overlap can be expected in these models. Furthermore, experts may disagree on various issues expressed in these models, and even if they do not, it is unrealistic to expect that the many numerical values extracted from them under dubious conditions and in different contexts will be consistent from one model to the other, from one expert to the other. It is therefore important to ensure that models merged to form one knowledge base maintain a certain degree of consistency and soundness. The loading of external files (to create a particular KB) involves basically the same incremental process described above for the knowledge acquisition process, but the statements are read sequentially from the external files. The necessary control information is associated with each statement in the external files, allowing it to be connected properly in the inference network and the Matcher is used to analyze the relationships between the most recently entered statement and those entered so far, independently of what model they belonged to. The measures described above can thus be taken in exactly the same fashion and ensure probabilistic consistency of the inference network.

2.1.1.3. Aid in Designing the Semantic Representation

Because statements in the KB can be arbitrarily complex, their semantic encoding is often entered manually during the KA phase. During a consultation, however, the user is allowed to volunteer information to the system, and a parser [38] creates the semantic representation corresponding to the volunteered statements. The kinds of statements that can be translated by the parser depend upon taxonomy contents and an external grammar. Whether the semantic representation of statements is entered manually or constructed by a parser, the KE needs to

determine if the resulting representation is adequate. He must ensure that it reflects the intentions of the DE in all situations that could occur during a consultation.

In the early development stages of Prospector, all the expressive power of semantic networks was invoked to encode each statement in the KB, and additional representation constructs were developed when needed to ensure a close correspondence between a statement and its semantic representation. It soon became clear, however, that in many situations (when considering the various uses Prospector made of the semantic representation of statements) the effort and the additional complexity introduced were not likely to be rewarded. In this section, we will discuss some of the representation issues we have been faced with in Prospector and show how the Matcher can be used, at least in part, to assist the KE in choosing an appropriate representation for statements in the KB.

2.1.1.3.1. Representation Considerations in Prospector

Each node in the inference network corresponds to a statement in the KB. Which semantic encoding is most appropriate for a particular statement will generally depend upon the following three considerations:*

(a) Its relationship with other statements in the KB
(b) The role it plays in the inference network
(c) The psychology of interaction between the user and the consultation system.

Because the Matcher can identify related statements and characterize the nature of the relationship between these statements in the KB, it can assist the KE not only in choosing an appropriate representation for each statement, but also guide him in deciding which statements may be combined to form larger statements or broken into smaller units. Using examples from Prospector, we will examine each of the three considerations in turn and show how they can influence the representation of statements in the KB.

---

* We are not concerned here with the issue of efficiency and storage requirements trade-offs of a particular representation scheme, but rather we are interested in speeding up the knowledge acquisition process without sacrificing the more important features we want supported in a consultation system.

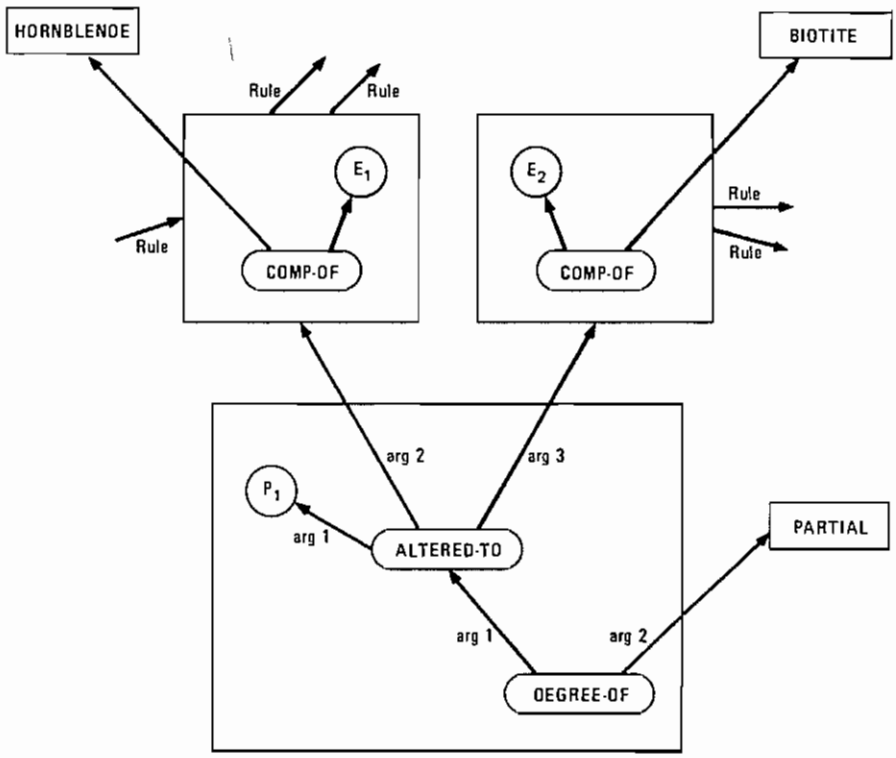


Figure 13 Alternative Semantic Representation of "partially biotized hornblende"

(a) Relationship with other Statements in the KB

Consider the statement:

S = "partially biotized hornblende."

One possible semantic representation of this statement was shown in Figure 3. If however, other statements exist in the KB that mention either "hornblende" or "biotite," and if these statements appear either as evidences or hypotheses of rules, then it may be reasonable to factor out these parts from S and represent S as in Figure 13. If, in addition, the process of "biotization" is mentioned elsewhere in the KB (maybe with a different degree), then the representation shown in Figure 14 may be more appropriate.

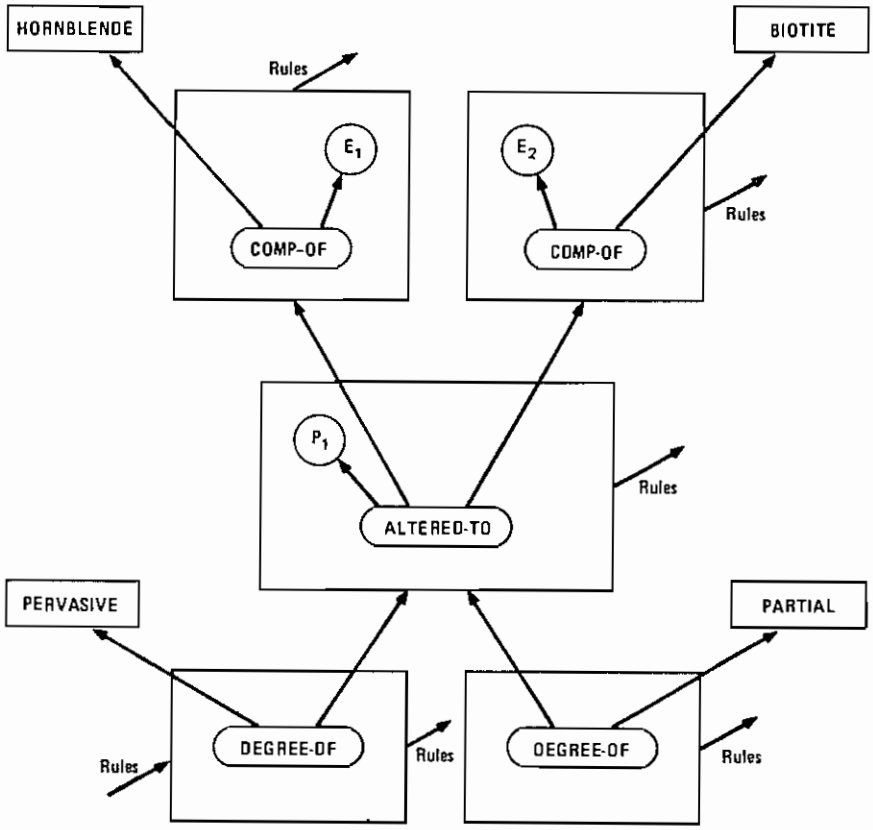


Figure 14 Alternative Semantic Representation of "partially biotized hornblende"

As another example of how statements can be grouped together or broken into smaller units, consider the statement

S = "rhyolite and dacite."

Two alternative semantic representations of this statement are shown in Figures 15 and 16. In Figure 15, the statement is encoded as one space describing the composition of two physical entities. In Figure 16, the statement is represented as two spaces connected by a logical construct (AND). If "rhyolite" and "dacite" always appear together in all other statements mentioning them, then the representation in Figure 15 may be most appropriate. If, on the other hand, either "rhyolite" or "dacite" is mentioned individually, then that of Figure 16 may be preferable.

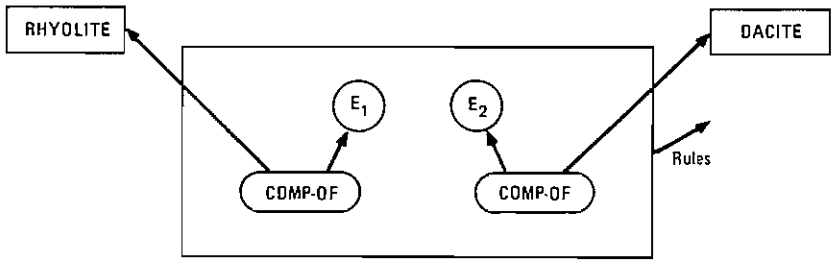


Figure 15 Semantic Representation of "rhyolite and dacite"

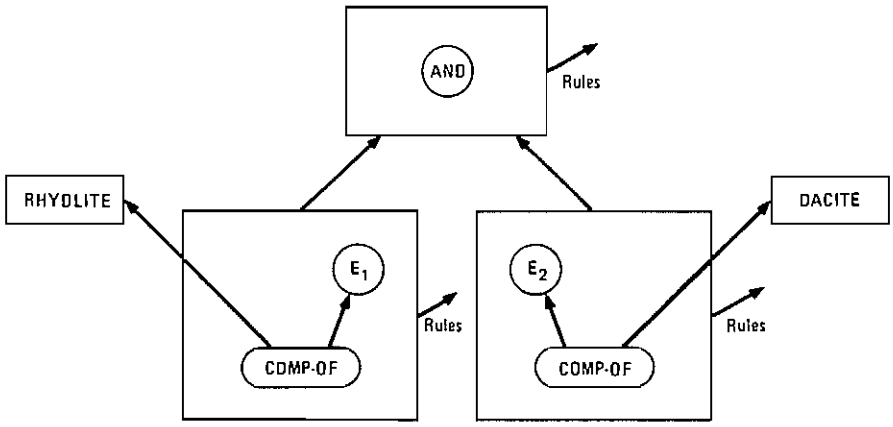


Figure 16 Alternative Representation of "rhyolite and dacite"

Clearly, no elaborate semantic encoding is necessary for a statement that is unrelated to any other statement in the KB and that does not contain any references to entries in the taxonomies. Because such a statement is unlikely to have a major effect on the consultation, a simple text-string representation would be adequate for most purposes. Furthermore, even though a statement S may be related semantically to other statements, only those portions of S and of those statements that are directly responsible for the semantic link need to receive a semantic encoding; a string representation would still be adequate for the remaining unrelated portions of the statements.

(b) Role Played in the Inference Network

Which of the several possible alternatives for each of the two statements in the above examples is finally chosen depends upon the nature of the relationships with the other statements in the KB and what role the statements play in the inference network. To determine how a particular representation will affect the deduction capabilities of the inference network and the course of the consultation, an analysis similar to that outlined in Subsection 2.1.1.1 must be undertaken.* For instance, if a link is found between two statements S1 and S2, and if S2

---

* Tools to assist in this analysis have been developed for the Knowledge Acquisition System of Prospector (see Chapter IV).

is involved in some rule R2, one of the alternative representations of S1 that is about to be added to the KB may result in R2's being invoked after S1 has been established during the consultation (because additional rules may have been created by the system to maintain the probabilistic consistency of the KB as shown in Subsection 2.1.1.1.1), while a different alternative may have no effect.

We should stress, however, that although a statement may be found semantically related to other statements in the knowledge base, a string representation of the statements involved may still be adequate if the relation discovered between the spaces is of no consequence to the inference network. For instance, suppose R1 is the rule: E1--->H1 and let E2 be the statement being added to the KB. Assume that E2 is a restriction of E1. If the likelihood ratios of R1 are such that only the absence of E1 affects the probability of H1 (LS=1, LN<1, for example), then any change of the probability of E2 during the consultation will have no effect on E1 (and hence no effect on H1). There would be therefore no need to associate an elaborate semantic representation with E2, at least as far as its relationship with E1 is concerned. We may, however, still need to remember for other purposes that the statements are related (see Subsection 2.2.2.2, for instance).

A special situation arises in Prospector where it is common to introduce several intermediate nodes (most often logical constructs) in the inference network with the sole purpose of holding and propagating intermediate changes during the consultation. For the statements corresponding to such intermediate nodes, even a string representation would be more than necessary. However, for several practical purposes and features of the consultation system (an explanation system, for instance), it is useful to associate a corresponding string representation with each node in the inference network.

(c) Psychology of Interaction

Any decision on how to represent any statement should be subject to the DE's approval because the final choice may not solely depend upon the semantic information conveyed by the statement and its role in the inference network, but may be forced by considerations related to the psychology of interaction with the user. The DE must take into account all situations that may occur during the consultation. One of the most important factors to be considered is often how the DE wants a question to be asked during consultation. He may feel that, in a particular section of the model, it is important (for geological reasons, say) that "X and Y" should be asked together while he may accept that they be asked separately in a different context. A long question, for instance, involving a conjunction of several items, may create confusion and ambiguity for a user. The DE may therefore sometimes decide to split a long statement into several units to be established individually in order to make the interaction more natural during the consultation session. The context construct in Prospector, a mechanism by which the KE can insert control information in the inference network to force some questions to be asked before others, may also dictate the choice of one

representation alternative before another, so as to avoid repetitions of similar phrases in consecutive questions during the consultation.

2.1.1.3.2. Semiautomatic Semantic Encoding of the KB

A semiautomatic procedure could conceivably be devised to encode all statements in the KB that would take into account at least two of the three representation considerations mentioned above.* Since the KB is being built incrementally, the most recently entered statement would be matched against the statements already in the KB, whereby the KE is asked to accept (or reject) the choice made by the procedure about what portions of the statement and of the statements it is related with, should be semantically encoded.

The main complication arises because not all the information is available at the time a statement is being added to the knowledge base to decide what portion (if any) of the statement should have a semantic representation. Two instances in Prospector where such complications occur involve:

(a) Volunteering Observations During the Consultation

In Prospector, the user is allowed to enter observations during a consultation session. For Prospector to take those volunteered observations into account and determine their effect on the existing knowledge base and on the course of the consultation, it must be able to find in the knowledge base those statements that are related to the volunteered statements. Thus, because only those statements that have a semantic representation are analyzed by the Matcher, we must anticipate at the time each statement is entered into the KB what related observations a user might reasonably want to volunteer during a consultation. If a related observation can be expected during a consultation, then the statement should have a semantic representation even though it is not related to any other statement currently in the KB.

(b) Multiple Experts and Models

The knowledge base is often created piecewise, perhaps by several experts each entering his own model for a particular subset of the domain knowledge. Not until all the inference networks encoding all models are merged into a single knowledge base is it possible to detect the interaction between statements from different models that may justify the semantic representation of the related statements (see Subsection 2.1.1.2). This means that the KE may have to review his decisions several times during the building of the KB.

---

* Note that we are concerned here only with what level of detail is needed to represent a statement that can be translated by a statement parser to a semantic representation. What the most refined representation looks like depends upon the representation primitives available, taxonomy contents, and the parsing capabilities of the language interface [38].

Let us examine a few simple steps we have taken in Prospector to alleviate these complications and which, in most cases, will help to avoid frequent updating of the KB's semantic encoding.

Use the Taxonomies as a Guide

The portion of a statement referring to basic concepts and primitive attributes of the domain should always have a semantic representation because those are likely to be volunteered observations during a consultation and occur in more than one model. These concepts and attributes coincide with entries in the taxonomies. In Prospector they include rock types, minerals, geological forms, geological ages, geological processes, as well as a set of primitive relations such as those described in Chapter II Section 2.

Develop and Extend the Taxonomies Independently of the Models

The taxonomies should be expanded to contain not only all the basic concepts and attributes of the domain mentioned in the statements that are currently in the knowledge base, but also those that may be relevant in the future (future models or volunteered observations). This approach increases the potential ability of the Matcher to discover relationships between statements. Suppose, for instance, that during a consultation, a user volunteers an observation about "pyrite" and that "pyrite" appears in the taxonomy of minerals as an element of sulfides, but is never mentioned in any model. Suppose now that "sulfides," on the other hand, is mentioned in some statement S, which therefore has a semantic encoding. Because "pyrite" appeared in the taxonomy, the Matcher will be able to establish through the hierarchical structure of the taxonomy that the volunteered observation is a restriction of "sulfides," which may result in modifications to the inference network. This link would not have been discovered if "pyrite" were not "anticipated" in the taxonomy.*

2.1.2. As a Search Feature – Accessing by Contents –

In determining an appropriate semantic representation of a statement, the DE must anticipate how a particular partitioning of the knowledge base will affect a consultation session involving his particular model. The KE, on the other hand, while preserving the intentions of the DE, will be concerned with possible interactions between several models or different parts of the same model in which case it may be necessary to modify the representation of the portions of the models involved. Typically, the DE or the KE will want to retrieve all the statements in the inference network that are related somehow to a particular statement he is about to add to the knowledge base so as to study the potential effects.

---

* A practical extension of the taxonomy is attained in Prospector by the use of synonyms for many geological concepts. This is particularly useful in the geology domain where different experts often associate different names with the same concept (specially rock types), depending upon how it is classified and used within their specialty.

Because development and testing of a KB typically extend over long periods, the KE cannot be expected to remember all the statements (or any labels assigned to them) that he, or another KE developing the same KB, has entered into the KB. The Matcher can be used as a search feature allowing the KE to access statements by specifying a partial description of their (semantic) contents. In effect, the Matcher-search feature (the TALKS command in RENE) will allow the KE to say something like:

"Now I want now to work on that portion of the inference network that deals with sulfide mineralization."

The search by content feature is accomplished by creating a semantic representation that corresponds as closely as possible to the partial description specified by the KE. The Matcher is then called to find any statement in the current knowledge base that links somehow to that partial description. The created semantic representation of the partial statement, together with the links computed by the Matcher, are maintained only temporarily and destroyed as soon as the search command is completed. Because this search feature uses the same programs as for volunteering information (see below), it has the same limitation with respect to the kind of partial descriptions that the system is able to parse and convert to semantic representations. Nevertheless, this capability has been shown to be a very powerful and convenient feature of the knowledge acquisition process.

2.2. Use of the Matcher During the Consultation Session*

During a consultation, the user is asked about evidences that may help to establish or rule out some hypothesis being evaluated by Prospector. To understand some of the features described below, let us examine how Prospector treats the user's answer. When the user is asked about an evidence E, he bases his answer on some observations E' and supplies a certainty measure C(E|E'). To maintain a uniform mechanism for propagating probabilities through the inference network, Prospector treats the user as if he were an evidence (a left-hand side) of a "dummy" rule that has E as a consequent (right-hand side). The rule strengths (likelihood ratios) for this "dummy" rule are computed, and are such that when the rule is actually applied, C(E|E') will correspond exactly to the user's answer. In computing these likelihood ratios, Prospector takes into account the prior probability P(E) as well as the effect of any other rules that may bear upon E (see Chapter II Section 3). Apart from its creation circumstance, the dummy rule looks like any other rule in the system. In particular, the inference network in which it is involved can be analyzed in the same fashion as described in Subsection 2.1.1.1 to ensure that it does not violate any consistency requirements imposed by restriction/generalization links discovered by the Matcher. Let us now describe the various situations in which this analysis is helpful during a consultation.

* Because controlled execution and testing of the inference network are supported in the knowledge acquisition system of Prospector, the features described under this heading are available during the KA phase as well.

2.2.1. As a Tool for Maintaining Consistency of the User's Answers:

2.2.1.1. Discovering Inconsistencies

As the user volunteers observations or answers questions asked by the consultation system, logical inconsistencies could result. Let E1 and E2 be two evidences such that E2 is a restriction of E1, and let P(E1) and P(E2) denote the current value of the probability associated with E1 and E2, respectively. Suppose that P(E1) is very low and that the user is asked about E2. If the effect of the user's answer is such that P(E1) < P(E2), there may be a contradiction. Two situations are to be considered:

(a) If P(E1) is low merely because it is the prior probability (not been affected yet in the consultation), or because of the effect of previously invoked rules, then Prospector will decide to believe the user's answer about E2 and uses the procedure described in Subsection 2.1.1.1.1 to add a rule having E1 as its consequent and such that P(E1) is increased to equal P(E2). A record is kept that the user's answer for E2 was the cause for the creation of this new rule. (This is needed in case the user decides to change some of his answers later during the session, see Subsection 2.2.1.2).

(b) If P(E1) is low because the user had previously been asked directly about E1 (or some other generalization of it) and had answered negatively, then the user's answers are really contradictory. In this situation no procedure can resolve the conflict. Instead, Prospector points out the inconsistency to the user and asks him to change his answer either to the earlier question about E1 or to the present question about E2.

There are several similar situations in which inconsistencies may be introduced during a consultation; they differ only in the way the contradictory statements involved occur in the inference networks. One or all those statements (or a negation of a statement) may have been volunteered by the user, answered directly by the user, or deduced by the system from other answers, volunteered observations or inference rules. All these situations can be discovered and treated in a similar fashion.

2.2.1.2. Changing Answers

A significant advantage of the Bayesian method used in Prospector for computing probabilities is the ease with which answers to questions can be changed without having to repeat all previous calculations. Basically, all that is required in changing an answer to a question about any evidence E is to change P(E) and propagate the results through the inference network.

The possibility of violating the restriction/generalization probabilistic constraints causes the only complication in this process. However, by keeping a record of how statements are related (as computed by the Matcher), the answer-changing program knows that it may also have

to change the probabilities of some of the related statements to maintain consistency. For instance, if the inference network contains the two rules E1 ---> H1 and E2 ---> H2, and the user gives a negative answer to a question about E1, P(H1) will be updated in accordance with the rule strengths associated with the first rule. In addition, if E2 is a restriction of E1, P(H2) must also be updated in accordance with the rule strengths of the second rule, as if a negative answer had been given for E2 as well. When the user then changes his answer to E1, the probabilities of both H1 and H2 should be automatically updated and propagated through the inference network.

By changing an answer, the user may contradict some of his earlier assertions, and changing these assertions may give rise to still further contradictions. This can confuse the user, but poses no problem for the recursive answer-changing program, which will make sure no contradictions are introduced before resuming a consultation.

2.2.2. As a Dialogue Management Tool

2.2.2.1. Mixed Initiative

Prospector can work in either of two modes --the consequent mode or the antecedent mode. In the consequent mode Prospector attempts to establish (or rule out) an hypothesis, and it traces backward through the inference network and searches for appropriate evidences to ask about. In the antecedent mode, the inference network is used in a forward direction to propagate the consequences of input information. The user has the option of requesting control any time during the consultation session to inform Prospector about facts he believes to be relevant and the consequences are propagated.

This capability to entertain a mixed-initiative dialogue mode distinguishes Prospector from most other expert systems currently being developed. Let us examine how the Matcher makes this possible.

2.2.2.1.1. Volunteering Information (antecedent mode)

The user volunteers information during a consultation session by typing simple declarative sentences, such as "there is a rhyolite plug surrounded by dacite dikes" or "there might be Precambrian quartz-monzonite." These sentences are parsed by a language interface, the LIFER subprogram [38]. The kinds of sentences that LIFER can analyze depend upon taxonomy contents and an external grammar. Currently, these two components allow the user to state the kinds of rocks, minerals, and alteration products that may be present on a prospect area as well as some common attributes such as geological ages and forms. A much simpler input parser is used at the beginning of a consultation, where the user is given the opportunity to volunteer in a "menu-like" form the rock types and/or minerals that are currently explicit entries in the taxonomy.* In addition, the user may state the degree of certainty with

---

* The current vocabulary comprises approximately 700 minerals and 300 rock types.

which he believes the volunteered statements. For each volunteered statement Sv, a corresponding semantic representation is created and the Matcher determines if Sv is related to any other statement currently in the KB.

If Sv is identical to an existing statement E, the probability P(E) is updated to agree with the user's statement and propagated through the inference network.

If Sv is a generalization of an existing statement E, then part of something that is significant to the KB has been volunteered. Similarly, if Sv is a restriction of an existing statement E, then all of something significant to the system has been volunteered. In both cases an updating of P(E) may be necessary* and the analysis of the inference network involving Sv and E may result in the modification of existing rules or the creation of new ones in order to maintain the KB consistent.

No updating of probabilities occurs if neither a restriction nor a generalization relation can be found between Sv and statements in the KB, and the volunteered statement will have no direct effect on the course of the consultation (see Subsection 2.1.1.1.1 and Subsection 2.2.2.1.2 below).

2.2.2.1.2. Control Strategy and Hypothesis Selection

The information volunteered by the user is often relevant to several hypotheses. We illustrate below how a simple scoring criterion is used in Prospector in conjunction with the information computed by the Matcher to select a reasonable goal hypothesis.

Each time the user volunteers a statement Sv about the prospect being evaluated, it is matched against statements in the KB. For the goal selection task, only those statements E corresponding to the following situations are of interest:

(1) Sv and E are identical
(2) Sv is a restriction of E
(3) Sv is a generalization of E
(4) Sv overlaps E.

Both first cases, where an exact match is found or the user actually volunteers something more specific than is contained in the KB, could result in updating of probabilities that could propagate through

---

* If Sv is a generalization of E, then P(E) will not be affected if the user's answer was positive, but will be reduced if the answer was negative. Similarly, if Sv is a restriction of E, then P(E) will not be affected if the user's answer was negative but will be increased if the user's answer was positive. In both cases when P(E) is unchanged, the situation is remembered when later in the consultation the system follows up with questions about the unmatched portions of the restricted statement (see Subsection 2.2.2.2).

the entire inference network. Unfortunately, situations in which the top-level hypotheses are affected are quite uncommon in practice.

Case 4 is the most common situation, but results in no probability changes at all. Case 3 is also relatively common but (unless Sv was a negative statement*) additional information is needed before P(E) can be updated and propagated through the inference network.

Thus, a typical consequence of volunteering is that several partial matches are found between volunteered evidence and the models, with but little change occurring in the probabilities for the top-level hypotheses. A strategy based on the selection of the most probable** hypothesis will not work because in most circumstances the changes resulting from the volunteered observations would be mostly local effects in dispersed portions of the models not affecting the top-level hypotheses.

After experimenting with various strategies, we believe that a more appropriate selection strategy should not ignore that the user said something relevant to the KB even though the probabilities of the top-level hypotheses are not affected.

The Jh criterion function currently used in Prospector provides a heuristic procedure for scoring a hypothesis that takes into account both the certainty of the top hypothesis of a model and the number and kinds of matches found between volunteered statements and the KB:

$$Jh(H) = 0.5\ [C(H) + Fh(H)],$$

where H is the hypothesis being scored and Fh(H) is some measure of the quality of match, that depends upon the following factors:

(a) The type of Matcher link (equality, restriction, generalization and overlap).

(b) The certainty of the volunteered statement Sv, (can be negative as well as positive).

(c) The size of the inference network describing H and containing the statement E, that was matched by Sv.

Without giving here all the mechanics of the Fh function, we will give a simplified outline of how it is computed.

We define an effective-certainty Ceff, for each statement E in the KB as follows:

1. If no volunteered statements link to E, then Ceff = 0.

2. Elseif there is an Sv identical to E, then Ceff = C(Sv).

3. Elseif all volunteered statements linked to E are restrictions of E, then
   let E+ be the one having the highest certainty;

---

* For instance, Sv = "No sulfides" and E = "pyrite"

** Meaning with highest probability

if C(E+) > 0, then Ceff = C(E+) else Ceff = C(E+)/2.

4. Elseif all volunteered statements linked to E are generalizations of E, then
   let E- be the one having the lowest certainty;
   if C(E-) < 0, then Ceff = C(E-) else Ceff = C(E-)/2.

5. Else
   determine the restriction space E+ with highest certainty and the generalization space E- with lowest certainty.
   If C(E+) > 0, then Ceff = max {C(E+), C(E-)/2}
   elseif C(E-) < 0, then Ceff = min {C(E-), C(E+)/2}
   else Ceff = [C(E+) + C(E-)]/2.

If several volunteered statements link to the same E in the KB, their contribution to Ceff are not accumulated and only the largest contribution is counted.

For each hypothesis H, the sum of the effective-certainty values of all statements involved in the inference network describing H, is taken as the measure of the degree to which the volunteered data linked to H. However, since larger inference networks are more likely to accumulate a larger sum merely because they possess more linkable statements, the sum is normalized by the number of linkable statements* in the inference network, N(H).

The function Fh is computed as

$$Fh(H) = \frac{1}{N(H)} \sum_{E \subset H} [Ceff(E)]$$

2.2.2.2. Psychology of Man/Machine Interaction

An important aspect of a consultation system is its interaction environment, probably the most important single feature that ultimately will decide whether or not an expert system will be accepted by the intended user community. Particularly important is that the user not feel ignored by the system and that the dialogues are not unilaterally monitored by the system. We have shown earlier how the Matcher supports a mixed-initiative dialogue environment, how Prospector is sensitive to observations volunteered by the user in selecting a goal hypothesis, and how the user is assisted in keeping his answers consistent in the course of the consultation. These features do, one way or another, affect the consultation. There is, however, another class of features that have no effect on the function of Prospector, but whose main purpose is to enhance the communication between the user and the expert system. As an example of such features that are supported by the Matcher, we describe

---

* The number of statements which actually have a semantic representation (and therefore have the potential of being matched by a volunteered statement) is used as a measure of size of the inference network.

below how Prospector computes some of the text it displays while asking questions.

At any time during the consultation, Prospector is trying to establish (or rule out) an hypothesis. The question that the control strategy selects to ask the user corresponds to some evidence E that is computed to be the most relevant in pursuing the current hypothesis. The selection is made on the basis of many factors, involving the current state of the consultation, the potential effects of anticipated answers, and so on. Before asking the question, however, Prospector must in some situations also make sure that the user understands the context in which the question is being asked. Suppose, for instance, that the user has volunteered "pyrite" earlier in the consultation (or has been asked about it directly by Prospector). Obviously, it would be unforgivable if the system were to ask later "Are there any sulfides?" since "pyrite" is a sulfide. If, however, the system were to ask about "pyrite veins," some kind of acknowledgment is needed to signal to the user that the system did not forget that "pyrite" was volunteered, but that it is investigating the possibility of that "pyrite's" being in the form of veins. The system has no difficulties, even in more complex situations (involving several levels of inference and logical deductions), in recognizing that the two statements are different and that the question is motivated. These differences, however, are often too subtle to be recognized by the human user, particularly if much time has elapsed between when the user volunteered the observation and when the question is being asked. He must therefore be reminded by Prospector about any facts that it thinks are relevant before asking the question. The information needed to recognize these facts are the links relating E to other statements in the KB and which have been computed by the Matcher and recorded at some earlier phase of the consultation or during knowledge acquisition. How these facts are presented to the user depends upon the current state of the statements involved. The state of a statement is determined by its certainty value and how that certainty was established. Initially the certainty C(S) of any space is zero, in which case the state of S is "undefined." When the certainty of S changes sufficiently, its state changes also.

Let A(S) denote the user's answer (on a -5 to 5 scale) to a question about S, and let C1, C2 and C3 denote the following conditions:

C1 -- the user was asked directly about S
C2 -- the user was asked about Sr, a restriction of S
C3 -- the user was asked about Sg, a generalization of S.

Then the state of S is defined by the following decision table:

State	C1	C2	C3	Certainty Range
Confirmed	T	-	-	$4 \leq A(S) \leq 5$
Suspected	T	-	-	$1 \leq A(S) < 4$
Doubted	T	-	-	$-4 < A(S) \leq -1$
Denied	T	-	-	$-5 \leq A(S) \leq -4$
Suspected-subset	F	T	-	$1 \leq A(Sr) \leq 5$
Doubted-superset	F	F	T	$-5 \leq A(Sg) \leq -1$
Suspected-rule	F	F	F	$1 \leq C(S) \leq 5$
Doubted-rule	F	F	F	$-5 \leq C(S) \leq -1$
Undefined	None of the above			

Let E be the statement being asked and let Eg be any generalization of E. Before the user is asked about E, the statements corresponding to each Eg whose state is neither undefined nor denied are displayed to the user preceded by a computed phrase as follows:

State of Eg	Phrase
Confirmed	You told me about ...
Suspected	You suspected ...
Doubted	I know you doubted ...
Suspected-subset	Your statements imply ...
Doubted-superset	I know there is reason to doubt ...
Suspected-rule	I have reason to suspect ...
Doubted-rule	I have reason to doubt ...

Thus, for example, if the user had previously answered "3" to a question about the existence of rhyolite, then before asking him a question about the existence of a rhyolite plug, Prospector would say

"You suspected rhyolite...
To what degree do you believe that there is a rhyolite plug?"

However, if the possible existence of rhyolite were the consequence of an inference rule, Prospector would say

"I have reason to suspect rhyolite...
To what degree do you believe that there is a rhyolite plug?"

Another set of leading phrases is used in formulating the question about E itself. Four of these phrases depend upon the state of E as follows:

State of E	Phrase
Undefined	To what degree do you believe E ?
Suspected-rule	I have reason to suspect E. What is your degree of belief about that?
Doubted-rule	I have reason to doubt E. What is your degree of belief about that?
Doubted-subset	To what degree do you believe E in general?

Another situation arises when E (being asked) is a restriction of Eg, but Prospector has enough information to warrant a message to the user. This would correspond to the state "suspected-superset" for E. In this situation, Prospector would say "I know about Eg in general; to what degree do you believe E in particular?"

Several other situations not described here could be handled in the same fashion. Although we believe that, whenever possible, the user should be shown evidence that the system listens to him, it must also be done intelligently and judiciously. The "state" mechanism described above can be extended to a variety of applications and tailored to suit the interaction environment needs of different users. For instance, the explanation system in Prospector, which may be invoked any time during the consultation to interrogate Prospector about its line of reasoning or examine the conclusions reached so far, also uses the state of the involved statements to compute sentences that are displayed together with the requested information. Clearly, these stock phrases are simple attempts to inform the user about the implications of his earlier statements. Although they are not necessary in any logical sense, they enhance communication between the user and the consultation system and often serve to make the logical processes of the expert system more evident.

VI CONCLUSIONS

1. On Generality of Knowledge Acquisition Tools

In the process of transferring the DE's knowledge to the computer program, one cannot expect to capture all the DE's expertise. How well the resulting knowledge base matches the DE's accumulated knowledge, (including his practical experience and other subjective decision-making processes) may depend greatly upon the representational formalisms chosen to encode the various elements of that knowledge. Depending on the domain and the purpose of the expert system, it may involve solving problems with many different "flavors," such as classification and diagnosis, planning and design, theory formation, simulation and prediction, interpretation and explanation, and so on. A representation that is appropriate for one flavor (or a combination) may be inadequate for another. In addition to chosing an appropriate representation, the KE must select an appropriate problem-solving mechanism, and it, too, depends on the domain and the particular tasks the expert system is expected to perform. Thus, in spite of the modularity of expert systems (clear separation of the rules from the rule interpreter and the global database), knowledge acquisition tools designed for a particular expert system cannot easily be used to assist in building another one. The collaboration of a full-fledged KE will be indispensible to tailor the combination of representation and deduction formalisms that are appropriate for each particular domain (i.e., that are able to represent and use the domain knowledge in ways that allow the expert system to attain a high degree of competence).

In attempting to remedy this unfortunate situation, recent research efforts in knowledge acquisition and expert systems have been directed at illustrating the generality of an expert system merely by removing the rules for a given domain and substituting rules for a different domain [31, 69]. However, every domain has its own peculiarities, and despite the good intentions of system builders, these inevitably influence the design or performance of the system. Changes are often required in many components of the system. Recognizing these facts, other researchers have recently begun developing "system-building" tools or languages for building expert systems.

The approach adopted in "system-building" systems such as AGE, for instance [47], is to provide the KE with a "library" of separate interconnectable preprogrammed modules corresponding to the most "popular" inference methods and representational formalisms. The KB can be represented either as sets of production rules or as frame systems (called units). For the inference engine, AGE supplies standard procedures for forward-chaining and backward-chaining, plus the possibility of implementing other strategies. A "blackboard system" [41] constitutes the standard global database. The KE can thus choose

some combination of these modules to be included in his expert system in much the same way a compiler selects subroutines from a system library and "links" them to produce an executable program. While such high-level languages for programming knowledge-based systems promise to shorten system implementation time, they do not deal with the problem of shortening the time required for knowledge acquisition and actual encoding, entering, and maintaining of the KB.

Our approach in KAS has been to view an expert system as a superposition of "layers of knowledge" where each layer further specifies the knowledge contained in previous ones or introduces new knowledge along some dimension. We have selected the first layer to contain general knowledge about networks. The subsequent layers contain further specification of the various kinds of networks, and knowledge about every component that constitutes the resulting expert system, including the inference procedures, the consultation system, and finally knowledge about the domain of application. The knowledge acquisition tools, such as the resident network editor and the bookkeeping system described in Chapter IV, are driven by the information contained in this layered structure, and their operation can be modified by simple declarations for modifying the information contained in the appropriate layer.

The advantage of this layered structure is that KA tools can be designed efficiently and with a high level of generality to assist in different phases of the KA process, where in each phase only some aspect of the knowledge base is of interest or available to the KE. It follows also that the same tools can be useful in building a broader class of expert systems because the "nuts and bolts" that constitute each layer can be replaced, new layers added and old ones discarded, affecting not only each component of the expert system but also their interactions. For instance, as we experimented with the design of expert systems in other domains (Appendix A), we observed that, although new constructs and mechanisms had to be invented and the capabilities of the Prospector system itself had to be extended to tackle situations introduced by the new domain, the necessary modifications to KAS were relatively trivial.

2. On Network Representations

The choice of networks to be our primary representation language in KAS is, of course, arbitrary. Our first layer of knowledge could contain knowledge about any other representational primitives or any other kind of knowledge and, at least conceptually, any number of layers can be added to our layered structure to fit a particular environment. It is our belief, however, that the assumption of a network representation of the KB has facilitated the implementation of many features supported in Prospector as well as in KAS.

3. On the Qualifications of the Knowledge Engineer

Although we did not aim to completely eliminate the KE from the KA process, we did aim to reduce as much as possible the requirements on his qualifications by making larger parts of the representation and inference formalisms of the expert system transparent to the KE. By integrating the description of the KB with its representation, and the editing processes with the execution and inference processes of the consultation system, the knowledge-engineering tools described in Chapter IV can assist the KE in most tasks described in Chapter III without requiring that the KE understand the mechanisms involved. Other tools (such as the bookkeeping system) watch over the KE's shoulder, providing protection against various kinds of errors so that he can concentrate with some degree of confidence on the KB design rather than on the mechanics of implementation.

By providing means to relate the statements in the knowledge base to each other, the semantic network matcher described in Chapter V has been an important instrument in maintaining the KB consistent in form and content as it grows. It also supports many of the features that constitute the AI contents of the Prospector system. We believe that the approach is a general one and can enhance the intelligent behavior of any knowledge-based system.

The modular organization of the knowledge KAS has about the various components of the expert system enables the KE to develop in a relatively short time the first version of a model (or a section of a model) that can be tested in the consultation system even though it is incomplete in some respect (Appendix B). This early prototype can be the concrete basis for subsequent improvements obtained during interviews with the DE and is often critical in unveiling fundamental problems particular to the domain for which the invention of new mechanisms may be required.

4. Summary

Starting from a Prospector system in which all domain knowledge was discarded, KAS provides the KE with an environment for building expert systems that is similar to that available in sophisticated programming language systems (such as INTERLISP) for the development of large computer programs. In fact, several ideas adopted in the design of KAS tools for the development of large knowledge bases were greatly inspired from techniques and tools developed or often recommended for providing powerful and intelligent support to the programmer [58, 59, 67]. KAS is of assistance in all phases of the knowledge-engineering process. We believe such aids are helpful in facilitating the KA task, but perhaps more importantly, they provide the framework for the design of better organized and more consistent knowledge bases. As the knowledge-engineering methodology becomes more stable in an expert system, more uniform results can be expected from its knowledge base, which can thus serve as the basis for uniform analysis and evaluation criteria for consistent decison-making.

We believe that the techniques employed in the KAS system are general ones and that they are steps toward the ultimate research goal of designing a KA system that acquires knowledge in direct interviews with the DE.

Appendix A

CASE STUDIES USING KAS

Appendix A

CASE STUDIES USING KAS

Several opportunities have presented themselves recently that enabled us to test the usefulness and versatility of the KAS environment. The most interesting experiences were those that involved domains other than the geology environment of Prospector. This appendix describes briefly two such experiences, and the detailed results will be found in forthcoming publications ([29, 35, 36]). The intent here is to illustrate that although new mechanisms had to be invented to handle particular situations presented by the new application, only minor extensions were required to KAS.

1. Spill Model

1.1. Background

During the last week of August 1980, a workshop was conducted in San Diego, California, assembling over 50 researchers in the field of expert systems. The goal of the workshop was to evaluate the knowledge-engineering tools and techniques currently used by the different expert system builders. In addition to providing a forum for sharing experiences, comparing and analyzing different expert system formalisms and knowledge-engineering styles, the results of the workshop were to be compiled into a reference book on expert systems and knowledge-engineering [35]. Six study groups were formed to work on different aspects of the subject: definition, knowledge acquisition, architecture, knowledge engineering, meta-cognition, and performance evaluation.

The task for each of the eight teams* in the KE group was to actually build an expert system using their particular formalisms and knowledge-engineering tools. The domain of application was known beforehand only to the organizers of the workshop (Rick Hayes-Roth, Don Waterman, and Doug Lenat) and a "mystery domain expert" would be available during the workshop for interview by all participants. Because most of the activities of the other study groups centered around the results and experiences of the KE group, only a few days were available for the development of the expert system.

1.2. The Problem

The "mystery domain expert" turned out to be two scientists (Carroll Johnsson and Sara R. Jordan) from the chemistry and computer sciences departments of the Oak Ridge National Laboratory (ORNL), who

---

* The following systems were represented: AGE [47], EMYCIN [69], EXPERT [75], HEARSAY-III [1], OPS4 [26], RLL [33], ROSIE [73], and KAS represented by R. O. Duda and the author.

were exploring various approaches to emergency response to oil and hazardous chemicals spills. Tasks that would be involved in such operation include:

* Discovery of spills, notification of proper authorities and containment crews
* Characterization of the spill and the probable hazards
* Emergency containment and secondary containment of the spill
* Spill source location and spill course determination
* Counter measures, cleanup, and disposal
* Coordination of operations and equipment
* Preventive measures.

A substantial document was distributed to all participants, containing relatively detailed descriptions of these tasks. The document also included maps of the drainage system, actual spill cases as well as hypothetical emergency situations, snapshots of tables and data bases in the large information system containing information about thousands of chemicals, buildings where hazardous chemicals are stored, procedures, regulations, and so on.

Because of the time limitation, it was suggested that only the source location task be encoded for the experiment.

1.3. Summary of the Solution

Since Prospector is particularly well adapted to weighing different hypotheses, the problem was formulated to suit the diagnosis/classification formalism. We selected the problem of giving advice to a possibly inexperienced on-scene coordinator about the best strategy to use in searching for the location of the source. (This would be a significant component of a more comprehensive expert system and is related to the general problem of setting priorities in a crisis situation). Four major strategies were identified and factors most relevant to choosing a strategy were found in the document. By in large, the skeletal version of the inference network corresponding to each strategy was organized and developed top down, working down to known terminal nodes that represented observational data.

We began using KAS as soon as an initial design for the first strategy was sketched. In fact, from that point on, I concentrated on entering and testing the inference network, while R. O. Duda was responsible for the design of the model. Duda researched the document, occasionally sought clarifications from the expert, and provided sketches of sections of the inference network to add to the system.

The mixed-initiative capability of Prospector proved particularly useful in this application, where the user is expected to have some initial information about the spill. Very little effort was required to demonstrate this feature: A small taxonomy of oils and acids was added to the knowledge base and a few strategically selected nodes in the inference network received semantic representations to enable Prospector

to respond to the mention of oils and acids, and to respond strongly to particular strong pollutants (such as PCB oils).

Although the knowledge base had obvious inadequacies and was small (about 50 spaces and 20 rules in the inference network and 40 nodes in the taxonomy of pollutants), compared with models developed previously for Prospector, plausible behavior was demonstrated in sample consultations very early in the development process. In particular, we quickly obtained a pilot version of the expert system that could respond to volunteered information, seek additional information in an efficient way, weigh and balance competing factors in reaching conclusions and explain its conclusions. In addition, several system features, such as recognizing synonyms, changing answers, tracing, and summarizing were obtained automatically.

1.4. Limitations of Prospector in the New Domain

Prospector does not have a natural way to monitor the activities of two or more agents or optimize resource allocation. Although the ability of Prospector to use itself recursively on a sub-classification problem can be used for this purpose, the more natural solution would be to allow the binding of variables in antecedents of rules to be used in the consequents (or more generally, the system should allow arguments to be passed between any two nodes in the inference network, in a fashion similar to that used to transfer control).

A new construct was invented to handle a particular aspect of crisis management--allow certain actions to be triggered if the state of the inference network satisfied certain conditions. The primary use of this construct during the experiment was to print out a message (when the certainty of a particular conclusion satisfied a specified test) to indicate that the appropriate action would have been triggered at this point. The modification to the Prospector system was trivial (requiring only the extension of the formal language for describing spaces to accept a new field, and a minor change to the control mechanism) and no modification to KAS was necessary.

2. Hydrology Domain

2.1. Background

As an offshoot of the Prospector project, a new project has began at SRI to develop an expert system to assist in the evaluation of multiple sources of information for predicting water flow. Currently, complex numerical simulation models are used to perform these hydrological evaluations. A large number of parameters, which reflect the climate, vegetation, geology, soil type, and other characteristics of the region of interest, are usually estimated by expert consultants and are used as input to the simulation program together with series of data (e.g., precipitation and temperature, runoff measurements from rivers etc.), collected from a variety of sources, including some with high degree of uncertainty (e.g., aerial imagery and eyewitness reports).

In the initial effort of the project we have been concerned with estimating the input parameters to the numerical simulation system. Using the techniques developed in Prospector, an attempt is made to capture in an inference network as much as possible of the experience and reasoning process of an expert consultant. Inference networks for about 15 parameters have been constructed so far.

2.2. New Mechanisms and Constructs Required by the New Domain

In addition to the probability associated with each node in the inference network (as in Prospector), a numerical value must be computed and associated with most nodes. Furthermore, because the user is allowed to give intervals as an answer to most questions asked by the system, a probability distribution must be associated with each node (rather than a single probability value). Thus, the result of a parameter estimation may be a range of values for that parameter and a probability distribution over the range of values.

There are various procedures for computing values. A formula may be supplied directly by the expert for computing the value of a node given that of its descendants or he may supply a table (n-dimensional) for looking up the value.

Major extensions to the inference and control procedures of Prospector were required and experimentation with various approaches is still going on, particularly as to ways of estimating and propagating probability distributions in the new types of inference networks. The extensions to KAS were relatively straightforward, requiring the declaration of new types of nodes such as "FORMULA," "TABLE," etc., and interfacing some of the properties of these new types with the bookkeeping system (declaring NEEDS-lists and defining type and value-checking functions).

3. Other Domains

Although not as extensively as for the two domains above, several experiments were conducted in other domains including

* Pattern recognition of hand-written characters (e.g., inference networks to distinguish between "B" and "8").
* Trouble-shooting and repair consultant for home appliances (a dryer).
* Evaluation of football teams for predicting outcome of game.

4. Improving the Efficiency of the KA Process

In devoloping most Prospector models a practical and more efficient strategy was employed than that described in Chapter III, whereby the model development process is viewed as two parallel tasks: the design of the model and its implementation. What we have called the KE all along can thus be viewed as actually being two different persons--a model

designer (MD) and a model implementer (MI). The MD interviews the DE and sketches pieces of inference networks, which are entered into the KB by the MI. Both the MD and the MI use KAS; the MD uses primarily features that assist him in extracting information from the DE (e.g., figuring out appropriate numerical parameters), whereas the MI uses features to enter and test the KB. As soon as the first skeletal version of the model is entered, it serves as the basis for further interviews with the DE, who will now be able to see immediately the effect of any modification to the model on the numerical behaviour and general flow of control of the model.

This strategy is particularly efficient when only limited time is available for interviews with the DE, as was recently demonstrated during the development of the latest Prospector model (the "Lacustrine Carbonaceous Siltstone Uranium" model).* The DE was available for only three days, so efficient use of time was therefore important. John Gaschnig was the MD and I was the MI. Before the DE departed, he was able to conduct plausible consultations with a model of about 150 spaces, and take with him questionnaires generated by the system to prepare for the fine tuning of the model.

Because inference networks have proven to be a suitable language for communicating with the (noncomputer scientist) DE, we believe that the sketching and drawing of networks will always be an important step in the knowledge-acquisition process. In our research aim to shorten the overall time for developing the knowledge base and minimize the qualifications of the KE, an obvious next step would be to build an interface between KAS and a graphics display system that would allow the KE to create and test inference networks as he interviews the DE. The possibility of such extension of KAS has been considered by the author in the early design stages of the system and taken into account in the implementation of most KAS features.

As the cost of bit map displays and the computing power to maintain them continues to drop, we believe that such an extension would greatly enhance not only the working environment in which expert systems are developed but also the overall cost efficiency of the knowledge-engineering process.

---

* As well as during the San Diego experience described above.

Appendix B

SESSION WITH THE RESIDENT NETWORK EDITOR (RENE)

Appendix B

SESSION WITH THE RESIDENT NETWORK EDITOR (RENE)

The actual computer transcript of a session with the REsident Network Editor (RENE) is given below. It is designed primarily to illustrate features of RENE in its current state of implementation. The syntax of the commands and the basic format of the interaction are explained in the Primer in Appendix C. In the transcript of the session, everything typed by the KE is underlined to distinguish it from system output. The numbers enclosed within parentheses at the right hand side of the transcript were inserted in the transcript file after the session was terminated and are keyed to the numbered comments following the transcript.

1. Scenario

The simple XYZ model pictured in Figure B-1 is a fictitious one used only for illustration. It has been tailored to contain several of the constructs most commonly used in the design of actual models in Prospector. In entering the model into the KB, we shall follow the design steps outlined in Chapter III Section 2.2, that is, we shall

* Build a skeletal model
* Specify the inference and control information
* Specify numerical parameters and other information needed by Prospector
* Test and revise the model.

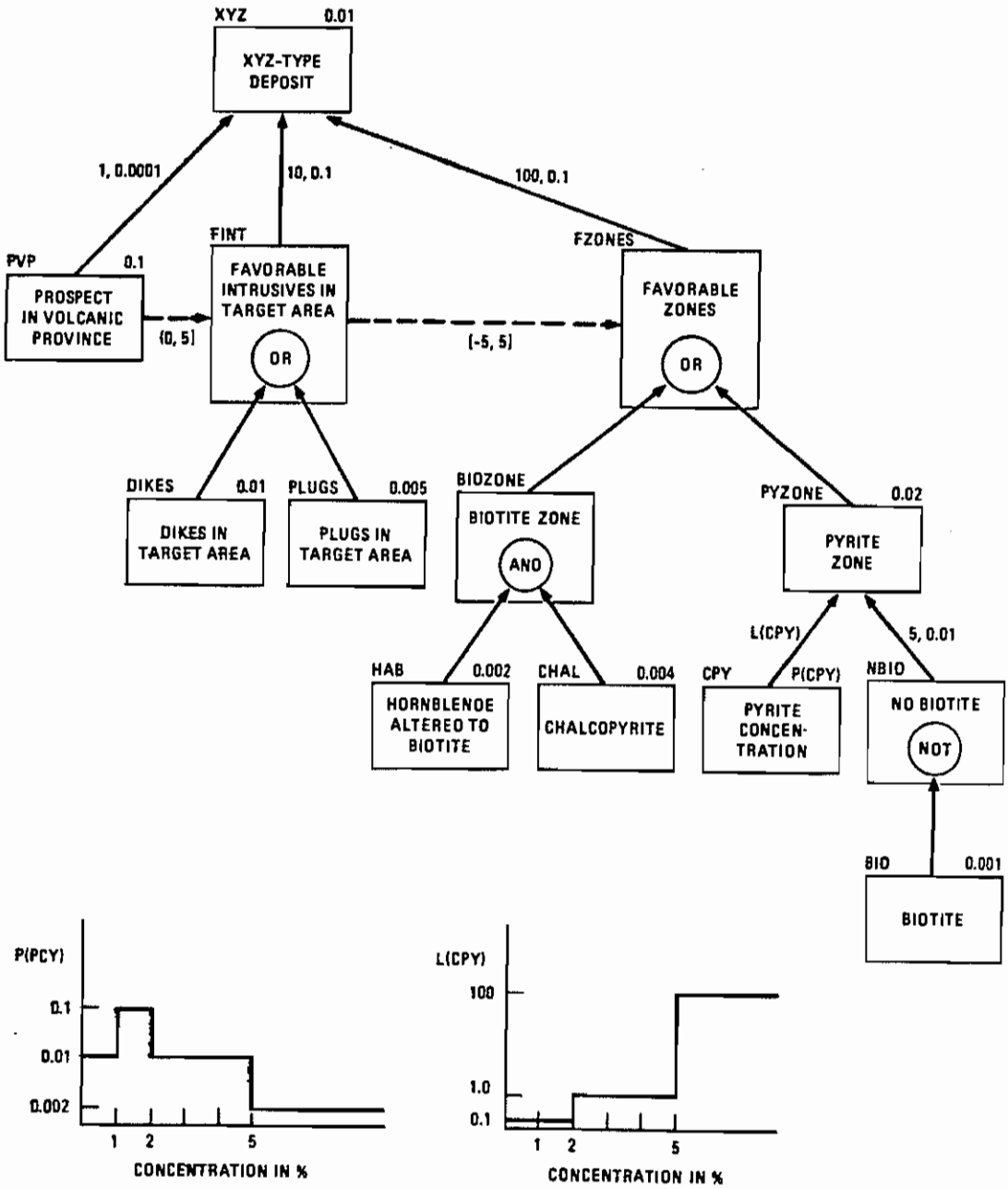


Figure B-1 The XYZ-model

2. Computer Transcript

RENE
The REsident Network Editor
of the PROSPECTOR Consultation System
SRI International
Version II
9-May-81 20:38:21

Do you want instructions? YES

RENE is a domain independent editor for creating and modifying networks. It knows particularly about the three types of networks used in the knowledge base of the Prospector system:

- Taxonomies
- Inference networks
- Semantic networks

Most commands for traversing or modifying the networks can be used with any type of network. There are, however, a few commands, that are useful only for a particular kind of network (e.g., the RUNFROM command for the controlled execution of a section of an inference network).

A "bookkeeping system" will keep track of the network structures being created and the user can interrogate the system about unfinished business with the NEEDS command. The COMPLETE command causes the system to enter a prompting mode in which missing parameter values are filled in under the supervision of RENE (checking for illegal parameter values).

The command interpreter is driven by a grammar that describes the syntax for the arguments of each command.
Type "?? cmnd" to interrogate the system about the syntax of a command "cmnd" or just type "??" (followed by a carriage return) if you want to see what commands are available.

>> ?? (1)

Available commands are:
1) A General Network Editor Command.
2) A PROSPECTOR-command.

Type "?? n" for more information about Option n.

```
>> ?? 1

Available commands are:

   ?                                  L         (Left)
   ??                               [M]LEAFS
   ADDCMND  (ADD CoMmaND)             LOAD
   ADDNEEDS                           MARK
   ATTN     (ATTentioN)               NEEDS
   BACKUP                             NN        (New Netobjects)
   CHFN     (CHange From-Nodes)       NTH
   CHTN     (CHange To-Nodes)       [M]OA       (Outgoing Arcs)
   CLEANOUTS                        [M]ON       (Outgoing Nodes)
[M]CN       (Create Nodes)           P         (Print)
   COMPLETE                           PP        (PrettyPrint)
   CONNECT                            PPROPS    (Print PROPertieS)
   CTYPE    (Connection TYPE)         PROPS     (PROPertieS)
   DELCMNDS (DELete CoMmaNDS)         PRPROPS   (PRopagate PROPertieS)
   DELNEEDS (DELete NEEDS)            PTREE     (Print TREEs)
   DESTROY                            QUIT
   DISC     (DISConnect)              R         (Right)
   DONE                               REMDUP    (REMove DUPlication)
   DOWN                               REMPROPS  (REMove PROPertieS)
   DUMP                               RENAME
[M]E        (Eval)                   RENAME.CMND
   EDIT                               SPM       (Set PM)
   EM       (Edit Models)             SWITCH
   FLAT     (FLATten)               [M]TN       (To Nodes)
[M]FN       (From Nodes)               UP
[M]IA       (Incoming Arcs)            WAIT
[M]IN       (Incoming Nodes)

Type "?? cmnd" for more information about the command cmnd.


>> ?? CONNECT                                                    (2)

CONNECT  ... syntax:
                [<ctype> ]<from.to.sequence>

>> ?? <CTYPE>

<ctype>  ... syntax:
               (e s de ds and-set or-set =s =)

>> ?? <FROM.TO.SEQUENCE>

<from.to.sequence>  ... syntax:
       ({[FROM ]<taxnodes> [TO <taxnodes> ]}
          {TO <taxnodes> [FROM <taxnodes> ]})
```

```
>> EM XYZ                                                          (3)
Creating the top space  SPACE-XYZ.XYZ for model  XYZ
 ===>> (XYZ)

** ??                                                              (4)

Available commands are:
        1) A General Network Editor Command.
        2) An Inference Network Command.
        3) A PROSPECTOR-command.

Type "?? n" for more information about Option n.

** ?? 2

Available commands are:

[M]ANTS     (ANTecedentS)              PM        (Print Models)
[M]ARCS                                PROPAGATE
   CHANGE                              PS        (Print Spaces)
   CONTEXT                             PSCTRL    (Print ConTRoL)
[M]CQTS     (ConseQuenTS)              PSCTX     (Print ConTeXts)
[M]CTXFOR   (ConTeXt FOR)              PSINF     (Print INFerence)
[M]CTXOF    (ConTeXt OF)               PSLINKS   (Print matcher LINKS)
   DCTX     (Destroy ConTeXt)          PSSEM     (Print SEMantics)
   ES       (Edit Semantics)           REDO
   ET       (Edit Taxonomies)        [M]RELS     (RELationS)
[M]EV       (EVidences)                RMODE     (Rule MODE)
[M]HYP      (HYPotheses)               RUN
[M]IEV      (Incoming EVidences)       RUNFROM
[M]IR       (Incoming Rules)           RUNUNDO   (RUN and then UNDO)
[M]LDOWNS   (Logical DOWN-Spaces)      SEM       (SEMantics)
   LUPS     (Logical UP-Spaces)        SHOW
[M]NODES                               SUMMARIZE
   NOTRACE                           [M]TALKS    (who TALKS about ...)
   NR       (New Rules)               TRACE
   NS       (New Spaces)              TRY
   OHYP     (Outgoing HYPotheses)     UNDO
[M]OR       (Outgoing Rules)          UNDOQ#    (UNDO Question#)
   PARSE                             LOADM     (LOAD Models)
   PICTURE                           DUMPM     (DUMP Models)


Type "?? cmnd" for more information about the command cmnd.


** ?? <CTYPE>

<ctype>  ... syntax:
                (RULES AND OR NOT PROC CONTEXT =*UNDEFINED =)
```

```
**  ?? <FROM.TO.SEQUENCE>

<from.to.sequence>  ... syntax:
      ({[FROM ]<spaces> [TO <spaces> ]}{TO <spaces> [FROM <spaces> ]})

**  ?? <SPACES>

<spaces>  ... syntax:
                  <space> [<spaces> ]

**  ?? <SPACE>

<space>  ... syntax:
               (A label for a node in the inference network)

**  CONNECT                                                          (5)

       [<ctype>]:
       FROM <spaces>: PVP FINT FZONES
       TO <spaces>: XYZ

**  CONNECT FROM HAB CHAL TO BIOZONE FROM CPY NBIO TO PYZONE

**  NEEDS ALL                                                        (6)

XYZ         (CTYPE PRIOR DESCRIPTION)
PVP         (CTYPE ASKABLE PRIOR DESCRIPTION)
FINT        (CTYPE ASKABLE PRIOR DESCRIPTION)
FZONES      (CTYPE ASKABLE PRIOR DESCRIPTION)
CPY         (CTYPE ASKABLE PRIOR DESCRIPTION)
NBIO        (CTYPE ASKABLE PRIOR DESCRIPTION)
PYZONE      (UPCONNECTION CTYPE ASKABLE PRIOR DESCRIPTION)
HAB         (CTYPE ASKABLE PRIOR DESCRIPTION)
CHAL        (CTYPE ASKABLE PRIOR DESCRIPTION)
BIOZONE     (UPCONNECTION CTYPE ASKABLE PRIOR DESCRIPTION)


**  CONNECT FROM DIKES PLUGS TO FINT FROM BIOZONE PYZONE TO FZONES

**  NEEDS PYZONE                                                     (7)

PYZONE      (CTYPE ASKABLE PRIOR DESCRIPTION)

**  RUNFROM XYZ                                                      (8)


1 -- To what degree do you believe that SPACE-PVP.XYZ ? -5
P(H|E') =  .1
```

```
2 -- To what degree do you believe that SPACE-DIKES.XYZ ? QQ

**  UNDO

**  CONTEXT PVP FOR FINT INTERVAL 0.0 5.0                          (9)

**  CTYPE RULES XYZ PYZONE

    RULE-PVP:XYZ.XYZ
    RULE-FINT:XYZ.XYZ
    RULE-FZONES:XYZ.XYZ
    RULE-CPY:PYZONE.XYZ
    RULE-NBIO:PYZONE.XYZ

**  CTYPE OR FINT FZONES

**  CTYPE AND BIOZONE

**  CONNECT NOT FROM BIO TO NBIO

**  NEEDS ALL

XYZ          (PRIOR DESCRIPTION)
BIOZONE      (ASKABLE PRIOR DESCRIPTION)
HAB          (CTYPE ASKABLE PRIOR DESCRIPTION)
CHAL         (CTYPE ASKABLE PRIOR DESCRIPTION)
CPY          (CTYPE ASKABLE PRIOR DESCRIPTION)
DIKES        (CTYPE ASKABLE PRIOR DESCRIPTION)
FINT         (ASKABLE PRIOR DESCRIPTION)
PLUGS        (CTYPE ASKABLE PRIOR DESCRIPTION)
FZONES       (ASKABLE PRIOR DESCRIPTION)
PYZONE       (ASKABLE PRIOR DESCRIPTION)
NBIO         (ASKABLE PRIOR DESCRIPTION)
PVP          (CTYPE ASKABLE PRIOR DESCRIPTION)
BIO          (CTYPE ASKABLE PRIOR DESCRIPTION)
PVP:XYZ      (LS LN)
FINT:XYZ     (LS LN)
FZONES:XYZ   (LS LN)
CPY:PYZONE   (LS LN)
NBIO:PYZONE  (LS LN)

**  COMPLETE ALL                                                   (10)

SPACE-XYZ.XYZ
          Prior = .01
          Description:  QN

SPACE-BIOZONE.XYZ
          Askable ?  QN
```

```
SPACE-HAB.XYZ
          Prior =  .002

SPACE-CHAL.XYZ
          Prior =  .004

SPACE-DIKES.XYZ
          Prior =  .01

SPACE-PLUGS.XYZ
          Prior =  .005

SPACE-PYZONE.XYZ
          Prior =  .02

SPACE-BIO.XYZ
          Prior =  .001


RULE-PVP:XYZ.XYZ
                LS =     1
                LN =     .0001

RULE-FINT:XYZ.XYZ
                LS =     10
                LN =     10
Invalid value for (LN)

RULE-FZONES:XYZ.XYZ
                LS =     100
                LN =     .1

RULE-CPY:PYZONE.XYZ
                LS =     LX

Intervals and priors for the evidence :  SPACE-CPY.XYZ
          Interval:  0.0 1.0
             Prior:  .01
          Interval:  1.0 2.0
             Prior:  .1
          Interval:  2.0 5.0
             Prior:  .01
          Interval:  5.0 99.0
             Prior:  .002
          Interval:  DONE

In interval  (0.0 1.0)       LS =     .1
In interval  (1.0 2.0)       LS =     .1
In interval  (2.0 5.0)       LS =     1.0
In interval  (5.0 99.0)      LS =     100.0
```

```
RULE-NBIO:PYZONE.XYZ
                LS =    5.0
                LN =    .01

** RUNFROM XYZ                                                    (11)

1 -- To what degree do you believe that SPACE-PVP.XYZ ? 5
P(H|E') =  .01

2 -- SPACE-CPY.XYZ ? QQ

** UNDO

** CONTEXT FINT FOR FZONES

** RUNFROM XYZ

1 -- To what degree do you believe that SPACE-PVP.XYZ ? 5
P(H|E') =  .01

2 -- To what degree do you believe that SPACE-DIKES.XYZ ? 5
P(H|E') =  .091743

3 -- SPACE-CPY.XYZ ? QQ

** UNDO

** COMPLETE ALL                                                   (12)

SPACE-XYZ.XYZ
          Description:  XYZ-type deposit

SPACE-BIO.XYZ
          Description:  there is biotite

SPACE-BIOZONE.XYZ
          Askable ?  QN
          Description:  biotite zone

SPACE-HAB.XYZ
          Description:  hornblende has been altered to biotite
```

```
SPACE-CHAL.XYZ
          Description:  there is chalcopyrite

SPACE-CPY.XYZ
          Description:  What is the pyrite concentration (in %)

SPACE-DIKES.XYZ
          Description:  there are dikes in the target area

SPACE-FINT.XYZ
          Description:  favorable intrusives in target area

SPACE-PLUGS.XYZ
          Description:  there are plugs in the target area

SPACE-PVP.XYZ
          Description:  the prospect is in a volcanic province

SPACE-FZONES.XYZ
          Description:  favorable zones

SPACE-PYZONE.XYZ
          Description:  pyrite zone

SPACE-NBIO.XYZ
          Description:  there is no biotite


RULE-FINT:XYZ.XYZ
                LN =      .1

**  NEEDS ALL

BIO          (CTYPE ASKABLE)
BIOZONE      (ASKABLE)
HAB          (CTYPE ASKABLE)
CHAL         (CTYPE ASKABLE)
CPY          (CTYPE)
DIKES        (CTYPE ASKABLE)
FINT         (ASKABLE)
PLUGS        (CTYPE ASKABLE)
PVP          (CTYPE ASKABLE PRIOR)
FZONES       (ASKABLE)
PYZONE       (ASKABLE)
NBIO         (ASKABLE)


**  RUNFROM FZONES                                              (13)
```

```
1 -- To what degree do you believe that the prospect is in a volcanic
province  ? 5
P(H|E') =  .02


2 -- To what degree do you believe that there are dikes in the target
area  ? 5
P(H|E') =  .02


3 -- What is the pyrite concentration (in %)  ? ??


Legal answers are:
1 ... The lower and upper bounds of an interval, or a single number.
2 ... A taxonomy name for an interval.
3 ... A PROSPECTOR-command.

Type "?? n" for more information about Option n.


3 -- What is the pyrite concentration (in %)  ? 4.6 5.8
What is your confidence in this estimate? 4
P(H|E') =  .0265661


4 -- To what degree do you believe that there is biotite  ? -2
P(H|E') =  .0645086


5 -- To what degree do you believe that hornblende has been altered to
biotite  ? SUMMARIZE

--------------------
        My certainty in

1. XYZ-type deposit
        is now 1.21547

Do you wish to see additional information?  YES
--------------------
I suspect that
1 - (* XYZ-type deposit) (1.21547)

There is one favorable factor:
1: 1.favorable intrusives in target area  4.99999

There is one positive  factor with neutral  effect that, if  negative,
could have been significant:
1: 2. You were sure that the prospect is in a volcanic province  5.0
                                    ** would have hurt if negative **
```

```
There is one uncertain factor whose score may be subject to change:
1: 3.favorable zones  .227085

For which of the above do you wish to see additional information? 3
--------------------
On a scale from -5 to 5, my certainty that
1.3 - (* favorable zones)
is now .227085

There is one favorable factor:
1.3: 1.pyrite zone  .227085                    ** controlling factor **

There is one factor that has not yet been considered:
1.3: 2.biotite zone  0.0

For which of the above do you wish to see additional information? NONE
Do you wish to see additional information about
1 - (* XYZ-type deposit) ?  NO

5 -- To what degree do you believe that hornblende has been altered to
biotite  ? QQ


**  PS CPY FZONES                                                    (14)

-----------------------------------------------------
space      CPY
    text   description  LX
/* What is the pyrite concentration (in %)*/
    semantics
           LOCATION      TARGET-AREA                                /
    inference
           prior in 0.0 1.0 use .01
                 in 1.0 2.0 use .1
                 in 2.0 5.0 use .01
                 in 5.0 99.0 use .002
           rules     consequents PYZONE    lhs  .1 .1 1.0 100.0
    control askable
    properties
           POSTERIOR .80027
           EXHAUSTED T
           ASKED T
-----------------------------------------------------
space      FZONES
    text   description
/* favorable zones*/
    semantics
           LOCATION      TARGET-AREA                                /
    inference
           prior .02
           logical definition   OR BIOZONE PYZONE
           rules     consequents XYZ       lh  100.0      .1
```

```
    control
           context of    FINT
    properties
           POSTERIOR .0645086

** UNDO

** SEM BIO                                                    (15)

SPACE-BIO.XYZ

!! ??

Legal answers are:
        1) DONE
        2) A phrase volunteering geologic information.
        3) A PROSPECTOR-command.

Type "?? n" for more information about Option n.

!! BIOTITE

!! DONE


** TRACE
OK

** SEM HAB

SPACE-HAB.XYZ

!! HORNBLENDE ATLERED TO BOITITE

   spelling-> ALTERED
              spelling-> BIOTITE

No delineation exists for relation ALTERED-TO
        Creating new taxonomy node ALTERED-TO under RELATIONS

Links:    to      SPACE-BIO.XYZ

!! DONE


Do you want to create the links? YES
        Inconsistent prior has been detected for  SPACE-HAB.XYZ (.002)
        The computed prior is:  .001
Do you want to change the prior of  SPACE-HAB.XYZ  to be  .001 ? YES
```

```
        Changing the prior for SPACE-BIOZONE.XYZ to .001

**  NOTRACE
OK

**  SEM PVP

SPACE-PVP.XYZ

!!  IGNEOUS ROCKS

!!  DONE


**  PS BIO HAB                                                    (16)

-------------------------------------------------------
space       BIO
    text    description
/* there is biotite*/
    semantics
           COMP-OF        E1            BIOTITE                  /
           LOCATION       TARGET-AREA                            /
    inference
           prior .001
           logical part  of NBIO  by NOT
    control bslinks  from  HAB
-------------------------------------------------------
space       HAB
    text    description
/* hornblende has been altered to biotite*/
    semantics
           COMP-OF        E1            HORNBLENDE               /
           COMP-OF        E2            BIOTITE                  /
           ALTERED-TO     E1            E2                       /
           LOCATION       TARGET-AREA                            /
    inference
           prior .001
           logical part  of BIOZONE   by AND
    control bslinks  to    BIO

**  PICTURE HAB

HAB
ALTERED-TO3-HAB arg2    E2-HAB.XYZ
                argl    E1-HAB.XYZ
                reltype ALTERED-TO

COMP-OF2-HAB arg2    BIOTITE
             argl    E2-HAB.XYZ
             reltype COMP-OF
```

```
COMP-OF1-HAB arg2     HORNBLENDE
             arg1     E1-HAB.XYZ
             reltype COMP-OF
```

** RUN (17)

```
XYZ  -- version 1
--------------------
```

If you want to give me some general information about the prospect, I can accept it at this time. However, since there is only one model to consider, it might be as effective merely to begin pursuing that model.
Do you want to volunteer any information? YES

This version of Prospector contains exploration models for the following types of ore deposits:

1 XYZ-type deposit

Different rock types are associated with different ore deposits. To help in selecting a model to pursue, it would be useful for me to know the types of rocks that are or might be present in your target area.

Do you have information about the lithology of the target area? YES

```
--------------------
```

Please name the principal types of rocks present in the target area.
(If you need instructions, type HELP; terminate by typing DONE.)

```
--------------------
1:TRACE
OK
--------------------
1:DACITE

SPACE-17
    semantics
          COMP-OF       E1           DACITE                    /
          LOCATION      TARGET-AREA                            /

              CERTAINTY: 5.0

Links:    to      SPACE-PVP.XYZ


 Changing the certainty of (DACITE)
       from 0.0 to 5.0
              Changing the score for:  SPACE-XYZ.XYZ  from 0.0 to 5.0

 Changing the certainty of (* the prospect is in a volcanic province)
       from 0.0 to 5.0
(DACITE) (5.0)
```

```
--------------------
2:DONE
--------------------
        Do you have information about significant minerals that are or
might be present in the target area?  YES

--------------------
        Please name these minerals.
        (If you need instructions, type HELP; terminate by typing DONE.)

--------------------
2:BIOTITE 3

SPACE-20
    semantics
          COMP-OF       E1            BIOTITE                          /
          LOCATION      TARGET-AREA                                    /

              CERTAINTY: 3

Links:    =       SPACE-BIO.XYZ

          from    SPACE-HAB.XYZ


 Changing the certainty of (* there is biotite)
        from 0.0 to 3.0
                Changing the score for:  SPACE-XYZ.XYZ  from 5.0 to 3.5

 Changing the certainty of (* there is no biotite)
        from 0.0 to -3.0

 Changing the certainty of (* pyrite zone)
        from 0.0 to -2.96939

there is biotite  (3)

--------------------
3:NOTRACE
OK
--------------------
3:DONE
--------------------
        Should you wish to volunteer additional information later
during the session, remember that you can interrupt at any time by
typing the VOL command.


3 -- To what degree do you believe that there are dikes in the target
area  ? 5
```

```
4 -- What is the pyrite concentration (in %)  ? 4.8 6.3
What is your confidence in this estimate? 3


You suspected that:
there is biotite  (3.0)


5 -- To what degree do you believe that hornblende has been altered to
biotite  ? 5
   !! This certainty seems inconsistent with your previous statements:
        2 -- there is biotite  (3)

Do you want to change the certainty for Statement 5 ? NO
New certainty for Statement 2 = ? 5


6 -- To what degree do you believe that there is chalcopyrite  ? 5



I have nothing more to ask about this hypothesis.
--------------------
I suspect that
H - (* XYZ-type deposit) (4.54503)

There are two favorable factors; in order of importance:
1.favorable intrusives in target area  4.99999
2.favorable zones  4.99995

There is one positive  factor with neutral  effect that, if  negative,
could have been significant:
3. I have reason to suspect that
                the prospect is in a volcanic province  5.0
                                  ** would have hurt if negative **


For which of the above do you wish to see additional information?
(Type ? for available options)  2
--------------------
On a scale from -5 to 5, my certainty that
2 - (* favorable zones)
is now 4.99995

There is one favorable factor:
2: 1.biotite zone  4.99995                          ** establishes 2 **

There is one factor that would have been unfavorable, had no favorable
factor existed to override it:
2: 2.pyrite zone  -4.90776
```

```
For which of the above do you wish to see additional information? NONE
Do you wish to see additional information about
H - (* XYZ-type deposit) ?   NO

**  UNDO

**  TALKS BIOTITE                                                    (18)

BIOTITE
Links:     =        SPACE-BIO.XYZ

           from     SPACE-HAB.XYZ


**  ATTN HAB
 ===>> (HAB)

**  UP
 ===>> (BIOZONE)

**  CTYPE RULES BIOZONE

    RULE-HAB:BIOZONE.XYZ
    RULE-CHAL:BIOZONE.XYZ

**  COMPLETE RULES

RULE-HAB:BIOZONE.XYZ
                LS =     50
                LN =     .01

RULE-CHAL:BIOZONE.XYZ
                LS =     40
                LN =     .01


**  COMPLETE SPACES

SPACE-BIOZONE.XYZ
          Askable ?  QN
          Prior =   .02

**  P                                                                (19)
(BIOZONE)

**  R
 ===>> (PYZONE)

**  UP
 ===>> (FZONES)
```

```
** NODES
(FINT DIKES PLUGS BIOZONE HAB CHAL PYZONE CPY NBIO BIO)

** LEAFS
(PVP DIKES PLUGS HAB CHAL CPY BIO)

** PTREE

FZONES or BIOZONE rules HAB
                        CHAL
          PYZONE rules CPY
                       NBIO not BIO


       ctx FINT or DIKES
                   PLUGS
               ctx PVP


(FZONES)

** RENAME PYZONE UNALTZONE                                          (20)

** PS FZONES CPY

-------------------------------------------------------
space      FZONES
    text   description
/* favorable zones*/
    semantics
           LOCATION      TARGET-AREA                                /
    inference
           prior .02
           logical definition   OR BIOZONE UNALTZONE
           rules     consequents XYZ       lh  100.0      .1
    control
           context of    FINT
    properties
           POSTERIOR .0199995
-------------------------------------------------------
space      CPY
    text   description  LX
/* What is the pyrite concentration (in %)*/
    semantics
           LOCATION      TARGET-AREA                                /
    inference
           prior in 0.0 1.0 use .01
                 in 1.0 2.0 use .1
                 in 2.0 5.0 use .01
                 in 5.0 99.0 use .002
           rules     consequents UNALTZONE lhs  .1 .1 1.0 100.0
    control askable
```

```
** CN FOO                                                          (21)
(FOO)

** TRACE
OK

** SEM FOO

SPACE-FOO.XYZ

!! PRECAMBRIAN DACITE DIKES ALTERED-TO LATE-PROTEROZOIC BIOTITE PLUGS

Links:    to      SPACE-PVP.XYZ
                  SPACE-BIO.XYZ

          overlap SPACE-HAB.XYZ

!! CHALCOPYRITE AND PYRITE VEINS
Links:    to      SPACE-PVP.XYZ
                  SPACE-BIO.XYZ

          overlap SPACE-HAB.XYZ

!! DONE

Do you want to create the links? YES


** PS FOO

-------------------------------------------------
space      FOO
    semantics
           FORM-OF       E1           DIKE                    /
           COMP-OF       E1           DACITE                  /
           AGE-OF        E1           PRECAMBRIAN             /
           FORM-OF       E2           PLUG                    /
           COMP-OF       E2           BIOTITE                 /
           AGE-OF        E2           LATE-PROTEROZOIC        /
           ALTERED-TO    E1           E2                      /
           COMP-OF       E3           CHALCOPYRITE            /
           COMP-OF       E4           PYRITE                  /
           FORM-OF       E4           VEIN                    /
           LOCATION      TARGET-AREA                          /
    inference     control bslinks  to    PVP BIO

** QUIT
Do you want to dump the models? YES

SPACE-BIO.XYZ
SPACE-BIOZONE.XYZ
SPACE-CHAL.XYZ
```

```
SPACE-CPY.XYZ
SPACE-DIKES.XYZ
SPACE-FINT.XYZ
SPACE-FZONES.XYZ
SPACE-HAB.XYZ
SPACE-NBIO.XYZ
SPACE-PLUGS.XYZ
SPACE-PVP.XYZ
SPACE-UNALTZONE.XYZ
SPACE-XYZ.XYZ
XYZ.KAS.1

        This terminates this RENE-session

@<REBOH>RENE.TRANSCRIPT.7
```

3. Comments on the Computer Transcript

The following comments are keyed to the numbers in parentheses (n) inserted in the right-hand side of the computer transcript above.

(1) The KE invokes the on-line help feature "??" to find out what commands are available, and follows up with "?? 1" whereby a selection of general network commands is displayed.

(2) The KE interrogates RENE about the syntax for the CONNECT command. RENE displays a BNF-grammar description for the arguments to the command, indicating, for instance, that the connection type <ctype> is optional. The KE follows up by inquiring about <ctype>. Because the default assumption at the outset of a RENE-session is that a taxonomy network is being created (indicated by the prompting string ">>"), RENE shows the labels acceptable for taxonomy arcs (various kinds of element/subset arcs). The "=s=" indicates that if the KE does not specify a label, an "s" arc will be assumed as a default.

(3) The KE instructs RENE that he will be working on an inference network. This enables RENE to choose appropriate defaults for nodes and arcs that will subsequently be created. (A new prompting string "**" will be used from this point on to indicate the new interpretation that will be associated with these nodes and arcs.) Since no model named XYZ exists yet, RENE creates the topspace (root-node) for the new model and initializes the position marker to point to that topspace.

(4) The KE again uses the on-line help command "??" to interrogate RENE about additional commands available in the context of building inference networks. Observe in particular that RENE now expects different connection types (rules, logical connections, and context arcs) and that the syntax for <from.to.sequence> indicates that nodes will be represented as <spaces>.

(5) By not supplying arguments to the CONNECT command (the KE has typed a carriage return after the command name), the KE chooses to let RENE prompt him for the arguments. RENE uses the grammar for the CONNECT command to get the arguments. In this particular instance, the KE types a carriage return for the optional argument <ctype> because at this stage, he merely wants to create the skeletal structure of the inference network, and therefore supplies only the names of the nodes to be connected.

(6) After connecting a few additional nodes (see diagram for the XYZ-model), the KE interrogates the bookkeeping system about unfinished business.

* PYZONE and BIOZONE have not been connected yet.

* The connection type, the askability status, a prior probability, and a text description is needed for all nodes except for the root-node, which is defaulted to be unaskable.

(7) The KE completes the construction of the skeletal network and checks the NEEDS of the node PYZONE to see that the UPCONNECTION need has now been satisfied.

(8) The KE can already make a test consultation with the skeletal model created. RENE will assume reasonable defaults whenever possible. For instance, all terminal nodes will be assumed to be askable, and the label of a node will be used when no description is associated with that node. After answering the first question with a -5, the KE is not satisfied with Prospector's choice of the next question, which indicates that Prospector is attempting to establish the FINT node. He therefore quits the consultation (QQ), undoes any of its side effects (UNDO), and adds a context arc between PVP and FINT with an interval restriction of (0.0 5.0), meaning that Prospector should not attempt to establish FINT unless PVP is established with a positive certainty value.

(9) The KE now specifies the types of connections for some of the nodes in the inference network and adds a connection from BIO to NBIO because he wants the user to be asked about "biotite" rather than about its negation. The bookkeeping system now shows the needs (the likelihood ratios LS and LN) associated with the arcs that have been specified to be rules.

(10) The KE requests assistance in completing the unfinished business. The BS enters a prompting mode and systematically goes through the NEEDS-list of each incompletely specified node or arc. The KE types "QN" for "description" and "askable," which signals to the BS to defer asking about those needs and use the defaults instead. Observe that the BS does not ask about the prior for logical nodes because it can be computed from the priors of the descendent nodes. An invalid value of the likelihood ratios for the rule FINT:XYZ is detected and rejected (each of the two values is legal but the combination is incorrect: they cannot both be greater than zero). The KE indicates that the rule CPY:PYZONE is a quantitative rule by typing "LX," whereby the BS enters a secondary interaction to acquire the prior distribution for the evidence CPY and the likelihood ratio for each of these intervals.

(11) A consultation is attempted. Because the LN value assigned to the rule PVP:XYZ (0.00001) has the most potential of ruling out the XYZ hypothesis, Prospector

asks first about PVP and then proceeds to investigate the branch FZONES:XYZ because the LS value assigned to that rule has the most potential for establishing XYZ. The KE does not like the flow of control and quits the consultation to add a new context arc between FINT and FZONES to force FINT to be investigated before FZONES.

(12) Now that the KE is satisfied with the basic flow of control, he completes the needs that are still unspecified (description text and the rejected LN value for the rule FINT:XYZ). The NEEDS-lists show that needs are still missing for some nodes because a default value is assumed or because the BS does not know which nodes will remain terminal and which will be connected to from other nodes as the inference network grows.

(13) The KE is now able to conduct a reasonable consultation. He chooses to test the FZONES section, but because of the context arcs the consultation proceeds as if it were started from the top hypothesis XYZ. After each question, RENE prints the current value of the probability P(H|E′) of the hypothesis being investigated (in this case FZONES). After question number 5 the KE invokes the explanation system (SUMMARIZE) to investigate more closely the behaviour of particular sections of the inference network.

(14) Before undoing the side effects of this consultation the KE prints the contents of the two nodes CPY and FZONES (included here to illustrate the formal language used in the external descriptions of inference networks).

(15) The KE proceeds now with the creation of the semantic representation for selected nodes of the inference network. The prompting string "!!" is used when semantic networks are being edited. The nodes BIO, HAB, and PVP are given a semantic representation corresponding roughly to the text description associated with each node. The trace has been turned on (by the TRACE command) for HAB to show the links discovered by the Matcher indicating that "HAB is a restriction of BIO". Because the taxonomies (assumed here to have been entered in a previous session) constitute part of the grammar used to create the semantic representations, simple spelling correction can be performed. Observe also that ALTERED-TO has been recognized as a relation and added to the appropriate taxonomy (the taxonomy used in this example does not know about alteration processes).

Having been notified about the links of the newly created representation (of HAB), the KE confirms to RENE to integrate that node with the current KB. Doing so, RENE discovers an inconsistency in the prior values assigned to the related nodes HAB and BIO (because HAB is a

restriction of BIO it is not allowed to have a prior probability (0.002) greater than that of the more general BIO (0.001). After the KE is forced to reconcile the inconsistency (by changing the prior of HAB to be 0.001), the BS automatically updates the inference network to reflect this change. In particular, the prior of the logical node BIOZONE must be recomputed.

(16) The KE examines the formal description of the two nodes BIO and HAB, which now contain the semantic representations. The PICTURE command shows the actual nodes and arcs in the internal representation of the semantic networks. It is intended primarily to be used in connection with a graphics interface (not implemented yet), and for debugging purposes.

(17) The RUN command initiates a complete Prospector consultation exactly as it would appear to the user if the XYZ model were included in Prospector. The user first volunteers some information: he is certain of the presence of a rock type (dacite), which links to PVP because it is an igneous rock, and somewhat less certain (3 in the scale of -5 to 5) of the presence of a mineral (biotite), which matches exactly with BIO and is related to HAB as before. The trace shows the effect of the volunteered information on the certainty values as they are propagated through the inference network.

Prospector does not have to ask now about PVP because the user has volunteered dacite and goes directly to DIKES. Observe also that before asking about HAB (question number 5), Prospector prints the leading phrase "you suspected that there is biotite," to indicate to the user that it is aware of his volunteering biotite but that it is now asking about something more specific, an alteration to biotite. The word "suspected" is selected because the user was not quite certain when he volunteered biotite. Had he been more certain, Prospector would have said "you told me..." instead.

The user answers "5" to question number 5. This is naturally detected as an inconsistency. He cannot be more certain about the special situation HAB than about (the more general situation) BIO. He is forced to resolve the inconsistency and selects to change the certainty of his volunteered statement. The consultation is terminated shortly therafter and the explanation system automatically invoked.

(18) The KE wants now to start modifying the inference network. He uses the TALKS command to access nodes in the inference network by their (semantic) content rather than by their label (useful in large knowledge bases or when the session is resumed at a later date and the KE

cannot remember all the labels). The two nodes that mention biotite in some fashion are shown and the KE moves to HAB (using the ATTN command) whereby the PM now points at HAB. The KE then retrieves the parent of HAB by using the UP command, changes the connection type of the incoming arcs to be RULES, and specifies the likelihood ratios of the newly created rules.

(19) The KE checks that the PM is still pointing at BIOZONE and uses the commands R (right) and UP to arrive to FZONES. The command NODES shows all the nodes "below" FZONES, LEAFS shows the leaf nodes of the inference network below FZONES, and PTREE displays the entire network structure below FZONES.

(20) The KE renames the node PYZONE to become UNALTZONE and examines the description of two adjacent nodes to verify that the new name is used.

(21) The KE creates a node FOO and gives it a semantic content. He does this in two steps resulting in the creation of four entities composed of dacite, biotite, chalcopyrite and pyrite. (This example illustrates the complexity of sentences that the current grammar can parse.) The links to other statements in the KB are displayed whenever the semantic content is modified. The Matcher discovers links involving PVP, BIO, and HAB as before.

The QUIT command terminates the session and permits the KE to save the created model in the external file XYZ.KAS.

Appendix C

PRIMER FOR THE RESIDENT NETWORK EDITOR (RENE)

Appendix C

PRIMER FOR THE RESIDENT NETWORK EDITOR (RENE)

This Appendix describes the features and commands of the REsident Network Editor (RENE) as it is currently implemented in the Prospector environment. Although it has several features to assist in the development of many kinds of network structures, RENE was primarily intended for the design and testing of inference networks and contains, therefore, a large amount of commands specially tailored for that task. We first describe the general features of the system (i.e., those useful for any kind of network) and then describe features that are particularly tailored to the development of inference networks.

1. Command Language

In the descriptions of the commands listed below, a BNF-like format is employed to describe the syntax of the arguments to the command:

* Upper case words are keywords (terminal symbols in the grammar).
* Lower case words enclosed in angle brackets <> are generic terms (nonterminal symbols).
* Optional items are enclosed in square brackets [].
* Alternatives are enclosed in parentheses ().
* Repetition of patterns is indicated by enclosing items in braces {}.
* Items between equal signs indicate default values assumed when the KE omits the corresponding argument.
* A sequence of similar items is indicated by the use of the plural form of the corresponding generic term such as:
  <items> ::= <item>[<items>].
* A quote sign (') preceding a generic term <g> refers to the symbol <g> itself (or any generic term appearing in the grammar that defines <g>. For instance, '<netobject> stands for the generic term <netobject> which can be <node> or <arc>, or for any of the other generic terms defining net objects such as <taxnode>, <space>, <rule>, <context-arc>, etc., (see Chapter IV Section 2.1.2).

Most generic terms used are self-explanatory. For instance, <space>, <rule>, <node>, <arc>, and <model> stand for a symbolic name associated with a space, rule, node, arc, and model, respectively. Similarly, <integer>, <certainty>, <question#>, <prop> and <string> stand for an integer number, a certainty measure (a number between -5

and 5), a question number (a sequence number assigned during a Prospector consultation to questions in the order they are asked and to observations volunteered by the KE), a name of a property associated with a net object, and a text string, respectively.

Other generic terms used include the following:

```
<netobject>        ::= (<node> <arc>)
<proplist>         ::= <prop> <value> [<proplist>],
```

The KE can interrogate the system (using the "??" command described below) about the syntax of any generic terms appearing in the description of a command and explore the grammar to any depth. The grammar displayed corresponds to the actual circumstances in which the command is being used. For instance, the syntax for the arguments of the CONNECT command is

[<ctype>]<from.to.sequence>

where <ctype> indicates the type of connection and <from.to.sequence> the nodes to be connected. If the KE is building a taxonomy, the syntax displayed for these two generic terms is

```
        <ctype>             ::= (e s de ds and-set or-set =s=)
  and   <from.to.sequence>  ::= ({FROM <taxnodes> [TO <taxnodes>]}
                                 {TO <taxnodes> [FROM <taxnodes>]})
```

If, instead, an inference network is being constructed, the displayed syntax will be

```
      <ctype>             ::= (RULES AND OR NOT PROC CONTEXT =UNDEFINED=)
  and <from.to.sequence>: ::= ( {FROM <spaces> [TO <spaces>]}
                                 {TO <spaces> [FROM <spaces>]} )
```

2. Name Conventions for Net Objects

A label is generally associated with every node and arc of the network being developed. It is part of the external description of the net object it is associated with, and is used in communicating with the KE. For taxonomies, the labels used are the names of the objects represented, such as "pyrite," "sulfides," or are directly related to the interpretation or function of the node or arc represented, (e.g., "e" (for element arc), "s" (for subset arc), "comp-of" (for a node corresponding to the comp-of relation), and so on). These labels do not necessarily uniquely identify a net object. Most arcs in a taxonomy, for instance, will be labeled "e" or "s." Internally, however, all net objects are uniquely represented. The KE often assigns labels of his choice to the net objects he is creating. This is particularly the case during the development of an inference network, where the labels associated with all the nodes are arbitrarily chosen mnemonics for the corresponding statements. Because of the large number and size of

models, as well as the participation of several KEs and DEs working on the design of the same model, naming conventions have been adopted in Prospector to assist the KE in assigning names and to avoid ambiguities in referring to the corresponding net objects.

The complete name of an object is composed of three parts: the type part (which indicates if the object is a space or rule), the short name of the object (which identifies the object within the model), and the model part (which identifies the model containing that object). In communicating with the KE, RENE uses only the short name. For example, in the model XYZ in Appendix B there is a rule between the two spaces named CPY and PYZONE. The complete internal names of these spaces are SPACE-CPY.XYZ and SPACE-PYZONE.XYZ, respectively. The short name of a rule indicates the antecedent and the consequent of the rule. The complete name for the rule CPY ---> PYZONE is thus RULE-CPY:PYZONE.XYZ.

These conventions allow the same name to be used for spaces and rules in more than one model. Names are unique within any model, and the bookkeeping system embedded in RENE makes certain that no ambiguities will arise. Note, however, that the names are completely independent of the internal representation and implementation of net objects. For instance, the type part of the name is not used to determine if an object is a node or an arc, and the short name of a rule is not used to retrieve the evidence or the hypothesis of the rule.

In all the commands described below, it will be sufficient to specify the short name of the objects referred to. The model part must be supplied if the KE refers to an object residing in a model different from that currently being developed, the "current model." The type part need never be specified unless the KE wants to assign names that do not comply with the standard conventions.

In the command descriptions listed below, we first give the name of the command followed by the syntax of its arguments (if any) and the mnemonics used in the name of the command when not obvious. The remaining text explains the operation performed when the command is executed. (The RENAME.CMND command can be used to change the name of any command to suit the preferences of the KE.)

3. Executive

The primary execution cycle of RENE corresponds to a loop in which it types a prompting character, reads a command line (terminated by a carriage return) typed by the KE and executes it. Different "prompting characters" indicate the kind of network currently being edited:

>>	for taxonomies
**	for inference networks
!!	for semantic networks

At the outset of a RENE-session, the prompting character displayed is ">>" indicating the default choice that a taxonomy is being edited. This and other default assumptions can be easily changed by simple declarations or switches. For example, the following three commands

will change the default assumption by the bookkeeping system as to what type of network is being edited:

EM <model>	(Edit Model)
ET <taxnode>	(Edit Taxonomy)
ES <space>	(Edit Space)

These commands do not restrict any of RENE's features, they merely instruct RENE as to the type of network that is primarily going to be worked on so that the appropriate defaults will be selected in subsequent operations. If, for instance, RENE is invoked with the ET command indicating that the KE is going to work on developing a taxonomy, he will be able to use commands for developing inference networks as well, but whenever a situation is ambiguous it will be interpreted in the context of the task of building a taxonomy.

Whenever the KE types a command without specifying arguments, RENE enters a prompting mode in which the grammar describing the syntax of the command is used to prompt the KE for appropriate arguments or assign defaults for missing arguments. In this prompting mode as well as at any time the system is expecting an answer, the KE can use the on-line help command "?" to interrogate the system about what type of answer it expects.

The KE terminates the session by typing DONE. He will then be talking to INTERLISP and can initiate a new RENE session* without losing the results of the previous one. However, if he wishes to retain the edit chain (see below), he should use the WAIT or the E commands described below to perform his INTERLISP computations.

The QUIT command terminates the entire run and asks the KE if the modifications to the KB should be saved on external files. A transcript file of the computer run is also written in the KE's file directory.

Whenever a command is executed that modifies the KB (either by creating new network structures or modifying existing ones, or assigning or modifying parameter or property values of net objects), the bookkeeping system is automatically invoked. In particular, the NEEDS-lists (and other bookkeeping information such as the default parameter and property values) are initialized or updated for all the net objects affected by the operation and as appropriate for each type of net object.

4. General Features and Commands Useful for Any Type of Network

4.1. Position Marker (PM)

The developer of large net structures will often want to perform various operations on several sections of the structure and switch his attention repeatedly among these sections. RENE maintains an edit chain, similar in spirit to that of INTERLISP, providing the bookkeeping mechanism necessary to support such operation.

---

* By evaluating the INTERLISP function call (RENE)

At any time during a RENE-session, a set of objects (perhaps only one) is the subject of the KE's attention. It could be a space being constructed, a list of incoming nodes in the taxonomy, and so on. The Position Marker (PM) is a cursor that always points at those objects currently being considered.

Each time the KE instructs RENE to shift its attention--either directly or by using commands that have such effect--the current PM is remembered in the edit chain for subsequent use. Thus KAS maintains a kind of "history list" of all the values of the PM accumulated during the session and the KE can retrieve any of these values and assign it as the current value of the PM.

Some commands described below will automatically modify the PM (such as the ATTN command), while others (such as all printing commands) will never alter it. However, for a variety of commands (particularly those that are useful for traversing the network) there are two versions of each command, one of which will modify the PM. This is indicated in the command descriptions below by preceding the name of the command with "[M]". For instance "[M]IN" indicates that there are two versions of the "IN" command (Incoming Nodes). If the KE wants only to find out which are the incoming nodes, he can use the "IN" command. If, in addition, he wants to actually "go there," he should use the "MIN" command, which will modify the PM.

In addition to moving around the network structure, the PM can be used to hold arguments to a command. For instance, if the KE wishes to perform several operations on the same object (or set of objects), he may assign that object as the value of the PM and omit it in the command. Thus,

> whenever applicable, in all commands, the current value of the PM is taken as a default value for the argument if no argument is supplied by the KE.

Each time the PM changes, RENE will notify the KE by printing the arrow "===>>" followed by the new value of the PM.

In all the command descriptions in this section, <pm> stands for a label that has been assigned by the KE to the value of a position marker. This can be done, for instance, with the MARK command described below. If <pm> is omitted, the current value of the PM is assumed.

The value of a position marker <pm> is always displayed as a list of items. These items may be net objects of any type (spaces, rules, taxonomy nodes or arcs). Since RENE does not know the intentions of the KE while he is manipulating the PM, it will not complain when a position marker is constructed that contains net objects of different types. The KE should be aware, however, that only one type of net object might be acceptable in arguments to several commands (if they accept the value of a position marker as a default argument).

The KE can manipulate the PM directly and use the commands described here to construct an appropriate argument for a command. This is particularly useful in cases where a value returned in the PM after executing a command is only slightly different from the argument required by another command.

4.2. Commands for Examining, Assigning, and Modifying the PM

P [<pm>] (Print)

Prints the value of the position marker assigned to <pm>. Uses the short-names format.

PP [<pm>] (PrettyPrint)

Same as P, but prints the complete internal names instead.

MARK <pm>

Assigns the current value of the PM to the label <pm> so it can be retrieved and used subsequently.

FLAT [<pm>] (FLATten)

"Flattens" the top level of the position marker referred to by <pm>. Several commands will make the elements of the PM lists rather than atomic elements. This is imposed by the network structure being considered and by the previous commands used to generate the PM. For example, suppose the current value of the PM is (S1 S2), where S1 and S2 are two nodes in an inference network. If S11 S12 and S21 S22 are the evidences of rules to S1 and S2, respectively, then the IN command (Incoming Nodes) will fetch the incoming nodes of the elements in the PM (the default argument to IN) and thus return ((S11 S12) (S21 S22)). For many applications the KE will prefer to see the flattened list (S11 S12 S21 S22). The KE may also want to bypass the subdivision, when only selected subelements of the PM may be needed in subsequent commands. For instance, if the KE intends to CONNECT the two nodes S12 and S22 to a node S0, the following command sequence will modify the PM so that it will contain only the desired elements:

```
MIN        (sets the PM to ((S11 S12)(S21 S22)))
FLAT       (flattens the PM to (S11 S12 S21 S22))
NTH 2 4    (sets the PM to (S12 S22),
                    the 2nd and 4th elements)
```

The from-nodes or the to-nodes may then be omitted in the CONNECT command: CONNECT TO S0

REMDUP [<pm>] (REMove DUPlication)

Removes duplication from the position marker referred to by <pm>. (Elements in the PM may have been arrived at via duplicate paths, which explains the duplication in the first place.)

4.3. Commands to Examine Net Objects and Travel Through Net Structures

ATTN <netobjects> (Attention)

Sets the current value of the PM to be the list of the specified objects. This is the "GOTO statement" of RENE, except that an arbitrary number of net objects may be selected. If any of the specified objects does not exist yet, the KE is notified and only the existing objects will be used.

Note: Since RENE does not know at this point what the KE is going to do with the fetched objects, it is the KE's responsibility to verify that these objects are of the type desired. Thus, for instance, it is perfectly correct to type

ATTN CPY CPY:PYZONE PYRITE

where CPY is an evidence, CPY:PYZONE is a rule, and PYRITE a node in the taxonomy.

SPM [<pm>] (Set PM)

Sets the PM to be the value of <pm> (previously assigned to some PM pointer, by using the MARK command, for instance). If <pm> is omitted, the value returned by the previous command will be used as the new value of the PM. (This is useful when the KE is traversing the network using commands like IN, ON, etc., and does not really know what values will be returned from these commands. He wants to change the PM only after having ensured that he reached the desired location of the network).

BACKUP [<pm>]

If <pm> is omitted, the first element in the edit chain becomes the new PM and that element is removed from the edit chain. Consecutive BACKUPs will thus get the KE back through the previous values of the PM. The argument <pm>, when specified, should be a marker (assigned by the MARK command, for instance) and the effect is that all elements of the edit chain up to and including that marker are removed from the edit chain, whereupon the specified marker becomes the new PM. Observe, however, that although an element is definitely removed from the edit chain, if it was assigned a label, it may be invoked for the duration of the entire session. (Because the label assigned to a PM is also used as a global variable to store the marker's value.)

SWITCH

Same as BACKUP, but does not lose the current PM. Instead the current PM and the first element in the edit chain change place, allowing the KE to switch back and forth between two values of the PM.

NTH <integers>

For each integer in in the sequence specified in <integers>, the nth element is extracted from the PM, and the current value of the PM is reset to include only those elements.

L [<node>] (Left)

Changes the PM to show the element that appears to the left of <node> (and at the same level), if a diagram were to be drawn picturing the network containing <node>. RENE arrives at that element by first ascending to the parent level and then choosing the descendant immediately preceding the element it started from. There may be ambiguous cases when there are several paths upwards from a given node, in which case RENE chooses the first one from the left of the tree structure it is examining. The KE can always go back to his starting point if dissatisfied with this choice. Although not precise, this command is very convenient for browsing through the network when the KE does not have a clear idea of where he wants to go.

R [<node>] (Right)

Works like the L command, but goes to the right instead.

UP [<nodes>]

Moves the PM up one level from each of the elements in <nodes>.

DOWN [<nodes>]

Moves the PM down one level from each of the elements in <nodes>.

[M]IA [<nodes>] (Incoming Arcs)

Shows the incoming arcs to each node in <nodes>.

[M]OA [<nodes>] (Outgoing Arcs)

Shows the arcs emanating from each node in <nodes>.

[M]FN [<arcs>] (From Nodes)

Shows the From-Nodes (origin) of each arc in <arcs>.

[M]TN [<arcs>] (To Nodes)

Shows the To-Nodes (destinations) of each arc in <arcs>.

[M]IN [<nodes>] (Incoming Nodes)

Shows the descendants of each node in <nodes>. Equivalent to executing IA followed by FN.

[M]ON [<nodes>] (Outgoing Nodes)

Shows the parents of each node n <nodes>. Equivalent to executing OA followed by TN.

[M]LEAFS [<nodes>]

Recursively tracks down the descendants of each node in <nodes> through the incoming connections and collects the leaf nodes (terminal nodes).

4.4. Commands to Create and Modify Networks

LOAD <filenames>

Loads external files containing descriptions of sections of the knowledge base. Each file may contain taxonomies (or taxonomy sections), and/or models (or model sections). The loading is incremental, whereby the contents of a file may redefine or

complete definitions contained in a previously loaded file. The bookkeeping system will keep track of missing information for each net object loaded from the file.

[M]CN <nodes> (Create Nodes)
Creates nodes with the indicated names if they do not already exist.

CONNECT [<ctype>] <from.to.sequence>
Creates connections between two or more nodes. <ctype> indicates the type of connection to be created. In inference networks, its value should be RULES, AND, OR, NOT, PROC, or CONTEXT. In taxonomies, it is the label associated with the arcs such as "e" or "s." If <ctype> is not supplied, RENE will create a connection with an appropriate default type. In an inference network, the "undefined" connection will be represented as a logical connection of the same kind as AND, OR, etc., and the bookkeeping system will remember that this connection was not fully specified. The <from.to.sequence> must be specified, at least partially. (If either the TO part or FROM part is left out, the current value of the PM will be used. If both the FROM and TO parts contain more than one element each, the cross product of all possible connections will be constructed.)

DISC <from.to.sequence> (DISConnect)
Deletes the connections specified by <from.to.sequence>.

CTYPE <ctype> [<nodes>] (Connection TYPE)
Sets (or changes) the type of the connection to each node in <nodes> to be <ctype>.

CHFN <node> (CHange From Node)
Changes the From-Node of each arc contained in the PM to be <node>.

CHTN <node> (CHange To Node)
Changes the To-Node of each arc contained in the PM to be <node>.

DESTROY [<netobjects>] (Destroy net objects)
Destroys all nodes and arcs specified in <netobjects>. (The bookkeeping system will note any sections of the network that are now disconnected from the rest of the network.

RENAME <netobject> <netobject>
Renames nodes and arcs.

4.5. Commands to Set and Modify Properties of Net Objects

PROPS <netobject> <proplist> (PROPertieS)
Sets (or changes) values for properties of <netobject> as specified by the property list <proplist>.

REMPROPS <netobject> <props> (REMove PROPertieS)
Deletes the properties for <netobject> as specified in <props> together with their values.

PRPROPS [<props>] (PRopagate PROPertieS)

Propagates the properties in <props> of all net objects contained in the PM. It assumes that a procedure INFERVAL(<node>, <prop>) is defined to compute the value of each property <prop> in <props> for a node of type <node>, given that of its descendents (Currently used in Prospector only for inference networks to force propagation of probabilities associated with selected nodes).

4.6. Printing Commands

DUMP <filename>[<netobjects>]

Writes the formal description of <netobjects> into the external file <filename>.

PTREE [<nodes>] (Print TREE)

Prints the network below <nodes> in a "prettyprinted" format showing the labels of the nodes and their connections.

PPROPS [<props>] (Print PROPertieS)

Prints the values of the properties in <props> of all net objects contained in the PM. All the properties and their values are printed if no argument is given.

4.7. Bookkeeping Commands

NEEDS [(<netobjects> '<netobjects> ALL)]

Shows what information RENE is still expecting the KE to specify for <netobjects>. The KE can also request to see the information still needed for a particular kind of net object by typing the corresponding generic name (e.g., <spaces> or <rules> in the inference network). If ALL is specified, all the information needed for the entire knowledge base constructed so far is shown.

COMPLETE [(<netobjects> '<netobjects> ALL)]

Enters a prompting mode in which the KE is prompted for the information needed as shown by the NEEDS command. The bookkeeping system will check for incorrect or inconsistent types and values in the KE's answers and set the values of the properties specified. Instead of answering the questions asked by the system, the KE can type one of the following:

- Q To defer answering this question now.
- QQ Defer answering this and all subsequent bookkeeping questions by exiting the COMPLETE command.
- QN Continue the prompting but defer inquiring about the need being currently asked about for the duration of this RENE session.

NN ['<netobjects>] (New Net objects)

Shows the net objects created so far during this session.

CLEANOUTS

As a result of some destructive editing operations, such as destroying nodes or connections between nodes, a section of the network may become disconnected from the main structure. This command shows such "floating" nodes that will have to be either deleted or reconnected by the KE before the session is over.

4.8. General-Purpose Commands

[M]E [<exp>] (Eval)

where <exp> stands for any legal INTERLISP expression. This is a call to the INTERLISP function EVAL, giving the KE a way to evaluate an expression without leaving the Executive. The ME version is particularly useful to "compute" the PM. If all the KE needs to do is some INTERLISP computation, he can use the WAIT command.

EDIT [<netobject>]

Calls the INTERLISP editor. The external representation of <netobject> is generated (in the formal language used for the description of the KB), and is edited as a regular list structure. The resulting expression is then parsed (by the PARSEFILE program) to generate the new internal representation. (This is mainly useful to the (knowledgeable) KE for fine tuning the semantic representations of spaces, in cases where the natural language interface is unable to produce satisfactory structures).

WAIT

Temporarily leaves the RENE executive but saves everything necessary for resuming the session where it was interrupted. After performing any INTERLISP computations, the KE can resume the session by typing "OK."

RENAME.CMND <oldcmnd> <newcmnd>

Changes the name of a command <oldcmnd> to be <newcmnd>. If <newcmnd> is already being used for an existing command, the KE will be notified and nothing will be done. Otherwise <newcmnd> will now have the same effect as <oldcmnd> and <oldcmnd> will not be recognized by RENE.

ADDCMND <cmnd> <fn> [<prfn> <cmndtype> <syntax>] (ADD CoMmaND)

With this command the (knowledgeable) KE can extend the RENE-command repertoire. Its use may require some LISP coding. Here <fn> is the name of the INTERLISP function to be applied to the arguments (if any) given by the KE when the command <cmnd> is to be executed, <prfn> specifies how the results are to be presented to the KE, <cmndtype> specifies one of several classes of commands, indicating, for instance, that the arguments are to be evaluated in a particular way or a side effect in executing the command (e.g., changing the PM or not), <syntax> specifies

the syntax of the arguments and is used to prompt the KE if any are omitted. (Similarly, there is a DELCMNDS command to remove commands from the repertoire).

ADDNEED/DELNEEDS (<netobject> '<netobject>) <needs>
Inserts (or removes) <needs> from the NEEDS-list of <netobject>. When used with '<netobject> it allows the (knowledgeable) KE to extend the bookkeeping system of RENE so that it will keep track (or ignore in the case of DELNEEDS) of the new needs included in <needs> (see Chapter IV).

?? [(<cmnd> '<generic-term> <option#>)]
This is an all-purpose command for interrogating RENE about the proper use of a command or the syntax of any generic term. RENE shows the syntax associated with the command <cmnd> or the generic term <generic-term> and the KE can explore the grammar to the desired depth. Whenever the KE is asked for information (such as in prompting mode while executing the COMPLETE command), he can use the "??" command with no argument. The system will then display in a "menu" format, the possible options preceded by a sequence number (<option#>), and the KE can follow up his inquiries about the desired option.

5. Special Commands for Inference Networks

Although the general commands described above are sufficient to perform most operations on inference networks, RENE supports a few additional commands (which can be viewed as macros) that are tailored to the task of developing models for Prospector. If, for instance, the KE wants to see all the outgoing arcs from a node in the inference network, then the OA command above will work fine. If he wants to see only those arcs that are rules, he may need to use additional commands to extract the desired information. The following commands will permit the KE to conveniently differentiate between the different types of arcs present in inference networks.

5.1. Commands to Examine Spaces and their Connections

[M]IR [<spaces>] (Incoming Rules)
Shows the rules that have their consequent in <spaces>.

[M]OR [<spaces>] (Outgoing rules)
Shows the rules that have their antecedent in <spaces>.

[M]EV [<rules>] (EVidences)
Shows the antecedents of the rules in <rules>.

[M]HYP [<rules>] (HYPotheses)
Shows the consequents of <rules>.

[M]IEV [<spaces>] (Incoming EVidences)
Same as IR followed by EV.

[M]OHYP [<spaces>] (Outgoing HYPotheses)
Same as OR followed by HYP.

[M]LUPS [<spaces>] (Logical UPSpaces)
Shows spaces of which the elements in <spaces> are a logical part.

[M]LDOWNS [<spaces>] (Logical DoWNSpaces)
Shows spaces that are the logical components of each element in <spaces>.

[M]CTXOF [<spaces>] (ConTeXt OF)
Shows spaces of which the elements in <spaces> are a context (i.e., that need to be established before <spaces> during a consultation).

[M]CTXFOR [<spaces>] (ConTeXt FOR)
Shows spaces for which the elements in <spaces> are a context (i.e., that cannot be established before <spaces>).

[M]ANTS [<spaces>] (ANTecedentS)
Combines LDOWNS and IEV; retrieves any spaces "below" <spaces>.

[M]CQTS [<spaces>] (ConseQuenTS)
Combines LUPS and OHYP; retrieves any spaces "below" <spaces>.

[M]TALKS <semantics>
Searches the KB for spaces having a semantic description that is related in any way to the semantic description specified by <semantics>. A scratch space is created with the specified semantics and the Matcher is called. The labels of the related spaces are displayed preceded by one of the links "=", "to", "from", or "overlap" to describe the relationship discovered by the Matcher. The KE can then use other commands to examine, manipulate or move to the retrieved spaces. Only simple statements involving attributes of physical entities such as composition (rocks and minerals), forms, geological age, and alteration processes, may be included in <semantics> (see SEM command below and examples in the computer transcript of Appendix B).

5.2. Commands to Create and Modify the Inference Networks

CONTEXT <space> <for.of.sequence>
Creates context arcs between <space> and the spaces S_i specified in the <for.of.sequence>, where

<for.of.sequence> ::=

{(FOR OF)<spaces>[INTERVAL <certainty><certainty>]}

FOR S_i means that <space> will have to be established before S_i in the consultation. OF S_i means that S_i will be established before <space>. In addition, the KE can specify an interval of certainty measures (two numbers between -5 and 5) which determines the certainty with which the context must be established (The CONTEXT command is equivalent to the CONNECT command (with <ctype>=CONTEXT), if the INTERVAL argument is omitted.)

DCTX <from.to.sequence> (Destroy ConTeXt arcs)
Deletes context arc specified by <from.to.sequence>.

5.3. Commands to Enter the Semantics and Other Contents of Spaces

SEM [<spaces>] (SEMantics)
For each <space> S_i in <spaces>, a prompting mode is entered in which the KE can type simple statements describing the semantic contents of S_i. If these statements can be parsed the corresponding semantic network representation is created and merged with the semantic structure constructed so far for S_i. As the semantic structure is being created, it is matched with the current knowledge base and the KE can see how the newly created semantic network is related to other statements. To conclude entering the semantics for each S_i, the KE must type "DONE." RENE asks for confirmation from the KE before integrating S_i with the existing KB. The bookkeeping system will assist in resolving any inconsistencies in the inference network discovered during this integration process.
Only relatively simple statements such as those used for volunteering information during the consultation are accepted. The current version of the grammar used to parse the statements can handle simple attributes of physical entities such as composition (rocks and minerals), forms, geological age, and alteration processes and can be extended by developing the corresponding taxonomies or expanding the grammar. To enter more sophisticated semantic structures, use the PARSE or the EDIT command.

PARSE <semantics>
Here <semantics> is given in the formal language used in the external description of the KB. (This is a call to the PARSEFILE program, a keyword-driven parser which converts the semantic description into the corresponding internal representation.)

LOADM <models>
Loads the external files containing the specified models from the KE's directory. (see DUMPM below) and creates the corresponding internal representation. The BS is invoked as if the structures being loaded were created with RENE commands.

5.4. Printing Commands

PS [<spaces>] (Print Space)
Prints the entire contents of the specified spaces in a format compatible with the PARSEFILE package. This command prints everything, including "?" and "why" texts, semantics, etc. If only selected information is needed, use the specialized commands below instead.

PSSEM [<spaces>] (Print Space SEMantics)
Prints the semantics in the PARSEFILE format.

PSCTRL [<spaces>] (Print Space ConTRoL)
For each of the spaces specified, prints the control information related to that space, i.e., the type of connections in which the spaces are involved, whether it has any contexts or other control constructs such as the BLOCK or PROC constructs.

PSINF [<spaces>] (Print Space INFerence)
Prints the inference information, namely, the prior probability, the incoming and outgoing rules, and their likelihood ratio values.

PSLINKS [<spaces>] (Print Space LINKS)
Prints the semantic connections between spaces that were generated by the Matcher indicating the relationships between them as described in Chapter V.

PSDESC [<spaces>] (Print Space DESCription)
Prints the description text of the spaces. This is the text used when the KE is asked about this space at run time (or for explanation under SUMMARIZE). The type of question is also printed if it is not a regular certainty question (i.e., Yes/No question, value question, map, etc.)

PS? [<spaces>] (Print ?-text)
Prints the elaboration text (supplied by the DE) that is a rephrasing of the question associated with the spaces, in greater detail.

PSWHY [<spaces>]
(Print WHY-text) Prints the explanatory text (supplied by the DE) attached to the specified spaces and explaining the geological backgrounds for the line of reasoning associated with the corresponding questions.

PM [<models>] (Print Models)
Prints the specified models in external format. The formal language described in Chapter III Section 3.1 shows the contents of the inference networks involved. All information associated with the inference networks is displayed, including control information, rule strengths, and the semantic contents of spaces. The printing of properties is driven by the value of the variable PROPERTIES, which can be reset by the KE to inhibit or include the printing of selected properties of spaces. The current model being edited is assumed if the argument is omitted.

DUMPM [<models>]
Dumps the specified models in external format into the KE's directory. The name of the file containing model "foo" will be "foo.kas." (If the model was incomplete, an additional file "foo.bkp" is generated, which contains the bookkeeping information accumulated so far. This file is not used by the LOAD command, which will regenerate the bookkeeping information as the model is being loaded).

5.5. Commands for Examining the Net Structure Inside a Space

PICTURE [<spaces>]

Prints the semantic structure showing the actual internal representation of the net structure in terms of nodes and arcs. (PSSEM shows the formal language description).

[M]NODES [<spaces>]

Shows all the nodes in the semantic representation of <spaces> that are "inside" the spaces (no external references).

[M]ARCS [<spaces>]

Shows all the arcs of the semantic representation of <spaces> that are inside the spaces.

[M]RELS [<spaces>] (RELationS)

Shows all the relations involved in the semantic representation of <spaces>.

5.6. Commands to Test and Run Models (Controlled Execution)

RUNFROM <spaces>

Starts a consultation session to establish <spaces> (useful for individually testing smaller sections of a model). The side effects (probability changes, askability status, etc.) will stay in effect after the command is executed.

UNDO

Undoes the side effects from the last RUNFROM (or RUN), executed.

RUNUNDO <spaces>

Same as RUNFROM, but will automatically undo the side effects at the end of the session.

REDO

Repropagates the answers from the last RUNFROM/RUNUNDO/RUN without repeating the questions. Typically, the KE executes a RUNUNDO (or a RUNFROM followed by an UNDO), edits the model, and wants to see the results of the change without repeating the consultation and having to answer all the same questions again.

RUN [<models>]

Simulates a complete consultation session exactly as it would be conducted by Prospector. It starts at the top spaces of the models currently in the KB. The KE can volunteer information, etc., just as in a regular consultation.

SUMMARIZE [(<models> <spaces>)]

Calls the explanation system. It can be used during a RUNFROM, RUNUNDO, or RUN commands to examine the current state of a consultation and investigate the reasons for the conclusions at all levels of the inference network.

SHOW <question#>

Allows the KE to examine his previous answers during the execution of any of the commands RUNFROM, RUNUNDO, or RUN.

CHANGE <question#>

Allows the KE to change answers to previous questions. Before effecting the changes, RENE notifies the KE if any inconsistencies are introduced by the new answers and enters a secondary interaction to assist the KE in resolving the problems.

UNDOQ# <question#>

Undoes the side effects of selected questions, those included in the sequence of question numbers <question#>, asked during the last RUNFROM, RUNUNDO or RUN.

TRACE/NOTRACE

Allows the KE to follow the intermediate conclusions made by Prospector after each question is answered by the KE while executing any of the controlled execution commands above (including the undoing commands).

SAVEMODE/BATCHMODE

SAVEMODE allows the KE's answers during a consultation (within RUNFROM/RUNUNDO/RUN), to be recorded on an external file so they can be used in a subsequent BATCHMODE session. The KE will be asked to supply the name of the external file and in the case of BATCHMODE, two procedure names. One will be used to automatically modify the answers fetched from the external file, the other, to compute an answer when none was available form the external file. The KE can answer "DFLT" when asked to supply the procedeures in which case the answers will not be modified and missing answers will be fetched at the terminal.

5.7. Commands to Assist in Determining Prior Probabilities and Rule Strengths

TRY <space>

This command provides a tool for experimenting with various values of the likelihood ratios for rules. RENE enters an interaction loop in which the KE is asked to give a number of rules for which <space> will be the consequent. He is then prompted for the needed parameters (priors probabilities, likelihood ratios) as well as for certainties of the antecedents. The KE can change all his answers (certainties, number of rules, etc.) as often as he needs, and each time RENE will compute the probability for <space>. The evaluations are performed in a "scratch" inference network and the KE can exit from this command by typing "QQ" to any question.

PROPAGATE [<spaces>]

Forces the propagation of the probabilities for <spaces>.

RMODE [(L C P)] (Rule MODE)

Changes the way the KE is asked about the rule strengths:

- L asks for the likelihood ratios LS and LN
- C asks for C(H|E) and C(H|~E)
- P asks for P(H|E) and P(H|~E)

The default is "L."

Appendix D

GLOSSARY OF ACRONYMS AND KEY TO MODEL DIAGRAMS

Appendix D

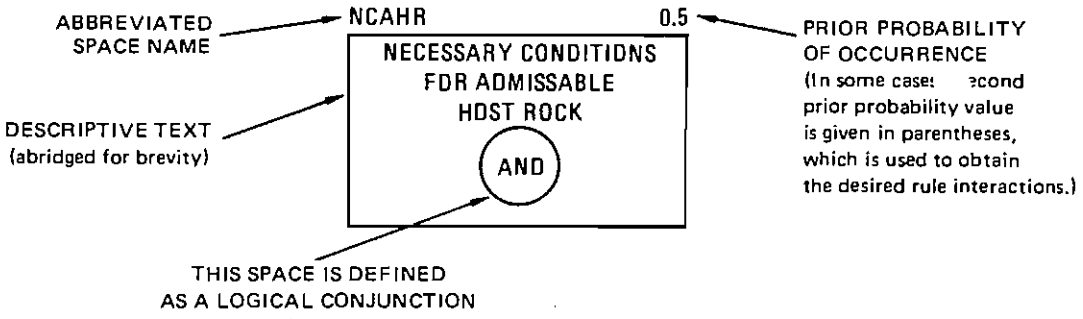
GLOSSARY OF ACRONYMS AND KEY TO MODEL DIAGRAMS

The following acronyms have been used throughout this report:

BS	Bookkeeping System
CS	Consultation System
DE	Domain Expert
DMS	Dialogue Management System
KA	Knowledge Acquisition
KAS	Knowledge Acquisition System
KB	Knowledge Base
KBS	Knowledge-Based Systems
KE	Knowledge Engineer(ing)
MD	Model Designer
MI	Model Implementer
PM	Position Marker
RENE	REsident Network Editor

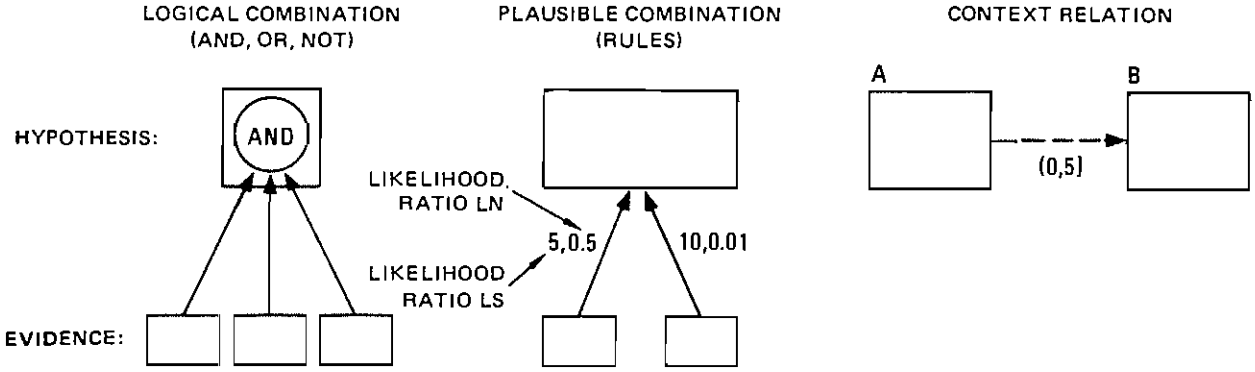
Schematic Key to Prospector Model Diagrams

ASSERTION SPACES:



NOTE: If box is dashed rather than solid, then its complete definition (including subnetwork, if any) appears on another page.

NETWORK LINKS:



INTERPRETATION:

In the case of an "AND" connection, all pieces of evidence must be present to establish the hypothesis. In the case of an "OR" connection, the hypothesis is established by any piece of evidence.

INTERPRETATION:

LS measures the degree of sufficiency or suggestiveness of the evidence for establishing the hypothesis. (A larger value of LS means greater sufficiency.) LN measures the degree of necessity of the evidence for establishing the hypothesis. (A smaller value of LN means greater necessity.) The value LS=1 (LN=1) indicates that the presence (absence) of the evidence is irrelevant to the hypothesis. For example, if LS>1 and LN=1, then the presence of the evidence is suggesstive of the hypothesis, but its absence does not lower the probability of the hypothesis.

INTERPRETATION:

Do not attempt to establish space B unless and until space A has been established with certainty greater than zero and less than or equal to 5. Context interval [-5,0) indicates A must have negative certainty before attempting to establish B. Context interval [-5,5] indicates simply that one should ask about A (regardless of the answer) before asking about B. Omitted context interval indicates (0,5].

REFERENCES

1. R. Balzer et al., "HEARSAY-III: A Domain-Independent Framework for Expert Systems," Proc. First Annual National Conference on Artificial Intelligence, pp. 108-110, Stanford, California (August 1980).

2. J. A. Barnett and M. I. Bernstein, "Knowledge-Based Systems: A Tutorial," Report No. TM-(L)-5903/000/00, Contract No. MDA 904-76-C-0343, System Development Corporation, Santa Monica, California (June 1977).

3. D. R. Barstow, "An Experiment in Knowledge-Based Automatic Programming," Artificial Intelligence, Vol. 12, pp. 7-119 (August 1979).

4. J. S. Bennett and R. S. Engelmore, "SACON: A Knowledge-Based Consultant for Structural Analysis," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 47-49, Tokyo, Japan (August 20-23 1979).

5. D. G. Bobrow and T. Winograd, "An Overview of KRL, a Knowledge Representation Language," Cognitive Science, Vol. 1, pp. 3-46 (January 1977).

6. R. J. Brachman, "What's in a Concept: Structural Foundations for Semantic Networks," BBN Report No. 3433, Bolt Beranek and Newman Inc., Cambridge, Massachusetts (October 1976).

7. R. A. Brooks, R. Greiner and T. O. Binford, "The ACRONYM Model-Based Vision System," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 105-113, Tokyo, Japan (August 20-23 1979).

8. J. S. Brown, R. R. Burton, and F. Zdybel, "A Model-Driven Question-Answering System for Mixed-Initiative Computer-Assisted Instruction," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-3, pp 248-257 (May 1973).

9. J. S. Brown, R. R. Burton, and A. G. Bell, "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An Example of AI in CAI)," Final Report, Contract F41609-73-C-006, Bolt Beranek and Newman, Inc., Cambridge, Massachusetts (March 1974).

10. B. G. Buchanan and T. M. Mitchell, "Model-Directed Learning of Production Rules," in Pattern-Directed Inference Systems,

D. A. Waterman and F. Hayes-Roth, eds., pp.297-312 (Academic Press, New York, 1978).

11. A. Bundy et al., "Solving Mechanics Problems Using Meta-Level Inference," in Expert Systems in the Micro Electronic Age, D. Michie, ed., pp. 50-64 (Edinburgh University Press, Edinburgh, Scotland, 1979).

12. W. J. Clancey, E. H. Shortliffe, and B. G. Buchanan, "Intelligent Computer-Aided Instruction for Medical Diagnosis," Proc. 3rd Computer Application in Medical Care, pp. 175-183 (1979).

13. L. S. Coles et al., "Methodologies for Heuristic Modeling," Final Report, SRI Project 3372, Contract 8719/62-8033-74WR, Stanford Research Institute, Menlo Park, California (April 1975).

14. R. Davis and B. G. Buchanan, "Meta-Level Knowledge: Overview and Applications," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 920-927, Cambridge, Massachusetts (August 22-25 1977).

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

16. T. G. Dietterich and R. S. Michalski, "Learning and Generalization of Characteristic Descriptions," Technical Report, Department of Computer Science, University of Illinois, Urbana, Illinois (1979).

17. R. O. Duda, P. E. Hart and N. J. Nilsson, "Subjective Bayesian Methods for Rule-Based Inference Systems," Proc. AFIPS 1976 National Computer Conference, Vol. 47, pp. 1075-1082 (1976).

18. R. O. Duda et al., "Development of a Computer-Based Consultant for Mineral Exploration," Annual Report, SRI Projects 5821 and 6415, SRI International, Menlo Park, California (October 1977).

19. R. O. Duda et al., "Development of the Prospector Consultation System for Mineral Exploration," Final Report, SRI Projects 5821 and 6415, SRI International, Menlo Park, California (October 1978).

20. R. O. Duda et al., "A Computer-Based Consultant for Mineral Exploration," Final Report SRI Project 6415 SRI International, Menlo Park, California (September 1979).

21. R. O. Duda, "The Prospector System for Mineral Exploration," Final Report, SRI Project 8172, Artificial Intelligence Center, SRI International, Menlo Park, California (April 1980).

22. R. Engelmore and A. Terry, "Structure and Function of the Crysalis System," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 250-256, Tokyo, Japan (August 20-23 1979).

23. E. A. Feigenbaum, B. G. Buchanan, and J. Lederberg, "On Generality and Problem Solving: a Case Study Using the DENDRAL Program", in Machine Intelligence 6, B. Meltzer and D. Michie, eds., pp. 165-190 (Edinburgh University Press, Edinburgh, Scotland, 1971).

24. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", in Artificial Intelligence, 2, pp. 189-208. (1971)

25. T. L. Fine, Theories of Probability (Academic Press, New York, 1973).

26. C. L. Forgy and J. McDermott, "OPS, A Domain-Independent Production System Language," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 933-939, MIT, Cambridge, Massachusetts (August 22-25 1977).

27. T. D. Garvey and M. A. Fischler, "Machine-Intelligence-Based Multisensor ESM System," Technical Report AFAL-TR-79-1162, SRI International, Menlo Park, California (October 1979).

28. J. Gaschnig, "Development of Uranium Exploration Moels for the Prospector Consultant System," Final Report, SRI Project 7856, Artificial Intelligence Center, SRI International, Menlo Park, California (March 1980).

29. J. Gaschnig, R. Reboh and J. Reiter, "Development of a Knowledge-Based Expert System for Hydrology Applications," Annual Report, SRI Project 1619, Artificial Intelligence Center, SRI International, Menlo Park, California (To appear June 1981).

30. H. L. Gelernter et al., "Empirical Explorations of SYNCHEM," Science, pp. 1041-1049 (September 9, 1977).

31. R. Goldberg and S. M. Weiss, "An Experimental Transformation of a Large Expert System Knowledge-Base," Working Paper, Department of Computer Science, Rutgers University, New Brunswick, New Jersey (March 1980).

32. I. Goldstein and S. Papert, "Artificial Intelligence, Language, and the Study of Knowledge," Cognitive Science, Vol. 1, pp. 84-123 (January 1977).

33. R. Greiner and D. B. Lenat, "A Representation Language Language," Proc. First Annual National Conference on Artificial Intelligence, pp. 165-169, Stanford, California (August 1980).

34. F. Hayes-Roth, "The Role Of Partial and Best Matches in Knowledge Systems," in Pattern-Directed Inference Systems, D.A-Waterman and F. Hayes-Roth, eds., pp. 557-574 (Academic Press, New York, 1978).

35. F. Hayes-Roth et al., "Building Expert Systems," F. Hayes-Roth, D. A. Waterman and D. Lenat, eds., (forthcoming 1982).

36. F. Hayes-Roth et al., "Tools for Building Expert Systems," Rand report, The Rand Corporation, Santa Monica, California, (forthcoming 1981).

37. S. Hagglund, "Contributions to the Development of Methods and Tools for Interactive Design of Applications Software", PhD Thesis, Software Systems Research Center, Linkoping University, Sweden (1980).

38. G. G. Hendrix, "LIFER: A Natural Language Interface Facility," SIGART Newsletter, No. 61, pp 25-26 (February 1977).

39. G. G. Hendrix, "Encoding Knowledge in Partitioned Networks," in Associative Networks--The Representation and Use of Knowledge in Computers, N. V. Findler, ed., pp. 51-92 (Academic Press, New York, 1979).

40. D. Lenat, "Automated Theory Formation in Mathematics," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 833-842, Cambridge, Massachusetts (August 22-25 1977).

41. V. R. Lesser and L. D. Erman, "A Retrospective View of the HEARSAY-II Architecture," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 790-800, MIT, Cambridge, Massachusetts (August 22-25 1977).

42. J. McDermott, "R1: An Expert in the Computer Systems Domain," Proc. First Annual National Conference on Artificial Intelligence, pp. 269-271, Stanford, California (August 1980).

43. M. Minsky, "A Framework for Representing Knowledge," in The Psychology of Computer Vision, P. H. Winston, ed., pp. 211-277 (McGraw-Hill Book Co., New York, 1975).

44. J. MOSTOW and F. Hayes-Roth, "Operationalizing Heuristics: Some AI Methods for Assisting AI Programming," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 601-609, Tokyo, Japan (August 20-23 1979).

45. A. Newell and H. A. Simon, Human Problem Solving (Prentice-Hall, Englewood Cliffs, New Jersey, 1972).

46. H. P. Nii and E. A. Feigenbaum, "Rule-Based Understanding of Signals," in Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, eds., pp. 483-501 (Academic Press, New York, 1978).

47. H. P. Nii and N. Aiello, "AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs, Proc. Sixth

International Joint Conference on Artificial Intelligence, pp. 645-655, Tokyo, Japan (August 20-23 1979).

48. N. J. Nilsson, Problem Solving Methods in Artificial Intelligence (McGraw-Hill Book Co., New York, 1971).

49. N. J. Nilsson, Principles of Artificial Intelligence (Tioga Publishing Co., Palo Alto, California, 1980).

50. S. G. Pauker, G. A. Gorry, J. P. Kassirer, and W. B. Schwartz, "Towards the Simulation of Clinical Cognition," American Journal of Medicine, Vol. 60, pp. 981-996 (June 1976).

51. H. E. Pople, Jr., et al., "DIALOG: A Model of Diagnostic Logic for Internal Medicine," Proc. Fourth International Joint Conference on Artificial Intelligence, pp. 848-855, Tbilisi, Georgia, USSR (September 1975).

52. R. Reboh et. al., "Study of Automatic Theorem-Proving Programs," Final Report, SRI Project 8776, Artificial Intelligence Center, SRI International, Menlo Park, California (November 1972).

53. R. Reboh, "Using a Matcher to make an Expert Consultation System Behave Intelligently," Proc. First Annual National Conference on Artificial Intelligence, pp. 231-234, Stanford, California (August 1980).

54. R. B. Roberts and I. P. Goldstein, "The FRL Primer," Memo 408, MIT Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1977).

55. J. A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," J. ACM, Vol. 12, pp. 23-41 (January 1965).

56. S. Rosenberg, "An Intelligent Support System for Energy Resources in the United States," Proc. Third Biennial Conference of the Canadian Society for Computational Studies of Intelligence, pp. 34-40, Victoria, British Columbia, Canada (May 14-16 1980).

57. E. D. Sacerdoti, et al., "QLISP: A Language for the Interactive Devolopment of Complex Systems," Proc. AFIPS 1976 National Computer Conference, pp. 349-356.

58. E. Sandewall, "Some Observations on Conceptual Programming", in Machine Intelligence 8, B. Meltzer and D. Michie, eds., pp. 195-204 (Edinburgh University Press, Edinburgh, Scotland, 1977).

59. E. Sandewall, "Programming in an Interactive Environment: The LISP Experience", in ACM Computing Surveys, Vol. 10, No 1, pp. 35-71 (1978).

60. E. H. Shortliffe and B. G. Buchanan, "A Model of Inexact Reasoning in Medicine," Mathematical Biosciences, Vol. 23, pp. 351-379 (1975).

61. E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN (American Elsevier, New York, 1976).

62. R. M. Stallman and G. J. Sussman, "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," Artificial Intelligence, Vol. 9, pp. 135-196 (1977).

63. J. L. Stansfield, "COMEX: A Support System for a Commodities Expert," Memo No. 423, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (August 1977).

64. M. J. Stefik, "Inferring DNA Structures from Segmentation Data," Artificial Intelligence, Vol. 11, pp. 85-114 (August 1978).

65. M. J. Stefik, "An Examination of a Frame-Structured Representation System," Proc. Sixth International Conference on Artificial Intelligence, pp. 845-852, Tokyo, Japan (August 1979).

66. W. Tetitelman, INTERLISP Reference Manual, Xerox Palo Alto Research Center, Palo Alto, California (October 1978).

67. W. Teitelman, "A Display Oriented Programmer's Assistant," in Int. J. Man-Machine Studies, Vol. 10, pp. 157-187, (1979).

68. M. H. van Emden, "Programming with Resolution Logic," in Machine Intelligence 8, B. Meltzer and D. Michie, eds., pp. 266-299 (Edinburgh University Press, Edinburgh, Scotland, 1977).

69. W. van Melle, "A Domain-Independent Production-Rule System for Consultation Programs," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 923-925, Tokyo, Japan (August 20-23 1979).

70. D. H. D. Warren and L. M. Pereira, "PROLOG--The Language and its implementation compared with LISP," Proc. Symposium on Artificial Intelligence and Programming Languages (1977); SIGPLAN Notices, 12(8); and SIGART Newsletter, no. 64, pp. 109-115.

71. D. A. Waterman, "Generalization Learning Techniques for Automating the Learning of Heuristics," Artificial Intelligence, pp. 121-170 (1970).

72. D. A. Waterman, and F. Hayes-Roth, eds., Pattern-Directed Inference Systems (Academic Press, New York, 1978).

73. D. A. Waterman, "User-Oriented Systems for Capturing Expertise: A Rule-Based Approach," in Expert Systems in the Micro Electronic Age, D. Michie, ed., pp. 26-34 (Edinburgh University Press, Edinburgh, Scotland, 1979).

74. S. Weiss, C. Kulikowski, and A. Safir, "Glaucoma Consultation by Computer," Computers in Biology and Medicine, Vol. 8, pp. 25-40 (1978).

75. S. M. Weiss and C. A. Kulikowski, "EXPERT: A System for Developing Consultation Models," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 923-925, Tokyo, Japan (August 20-23 1979).

76. S. H. Weiss, C. A. Kulikowski and B. Nudel, "Learning Production Rules for Consultation Models," Proc. Sixth International Joint Conference on Artificial Intelligence, pp. 948-950, Tokyo, Japan (August 20-23 1979).

77. J. Wieslander, "Interaction in Computer-Aided Analysis and Design of Control Systems". PhD Thesis, Dept. of Automatic Control, Lund Institute of Technology, (1979)

78. W. T. Wipke, "Computer Planning of Research in Organic Chemistry," in Computers in Chemical Education and Research, E. V. Ludena, N. H. Sabelli, and A. C. Wahl, eds., pp. 381-391 (Plenum Press, New York 1976).

79. L. A. Zadeh, "Fuzzy Sets," Information and Control, Vol. 8, pp. 338-353 (June 1965).

80. L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility," Fuzzy Sets and Systems 1, pp. 3-28 (1978).