Zeitschrift: Helvetica Physica Acta
Band: 59 (1986)
Heft: 1

Artikel: Knowledge engineering with expert systems
Autor: Humpert, B.
DOI: https://doi.org/10.5169/seals-115689

Helvetica Physica Acta
Vol. 59, 1986 21 - 87

0018-0238/86/010021-67$1.50+0.20/0

KNOWLEDGE ENGINEERING WITH EXPERT SYSTEMS *)

B. Humpert**)

Artificial Intelligence Group
HASLER Research Laboratories
CH-3000 BERN 14, Switzerland

and

Institut de Physique Nucléaire
Université de Lausanne
CH-1000 LAUSANNE, Switzerland

ABSTRACT

This paper examines a few typical methods for knowledge representation and demonstrates their application in a set of expert systems. We explain in particular the function mode of the systems/shells: MYCIN/EMYCIN, HEARSAY-II/III, PROSPECTOR/KAS, and finally we elaborate on their possible fields of application.

*) Invited Talk at the Meeting of the Swiss Physical Society, Bienne, 3-4 Octobre 1985.

**) also at CERN.

1. INTRODUCTION

A part of artificial intelligence research [1] involves the construction of systems which involve knowledge. A representation of knowlege is a combination of data structures and interpretive procedures that, if used in the right way in a program, will lead to "knowledgeable" behaviour. Work on knowledge representation in AI has involved the design of several classes of data structures for storing information in computer programs, as well as the development of procedures that allow "intelligent" manipulation of these data structures to make inferences. What type of knowledge is needed to be represented in AI systems? We can think of several kinds, such as:

- objects: knowledge in terms of facts about objects, their classes, categories or descriptions;
- events: knowledge about actions and events with a formalism indicating the time course of a sequence of events and their cause-and-effect relations;
- performance: knowledge about how to do things, the performance of skills;
- meta-knowledge: knowledge about what we know including knowledge about our own perfomance, as cognitive processors: our strengths, weaknesses, confusability, and so on.

Knowledge-representation schemes are not chosen at will, but they are subject to constraints allowing, for instance, for an easy way of reasonig. There is a set of different kinds of reasoning one might imagine such as for instance:

- formal reasoning, involving the syntactic manipulation of data structures to deduce new ones following prespecified rules of inference;
- procedural reasoning, using simulation to answer questions and solve problems;
- reasoning by analogy, which is a natural mode of human thought;
- reasoning by generalization and abstraction;
- and finally, meta-level reasoning.

Any construction of expert systems [2] is faced with this kind of questions, in particular so if the system is intended as a multi-purpose facility which allows for complex and involved tasks. One here might also be concerned with questions of efficiency which we however will not address. Expert Systems are distinct for three reasons: First, they perform difficult tasks at expert levels of performance. Second, they emphasize domain-specific problem-solving strategies. Third, they employ self-knowledge to reason about their own inference processes, and provide explanations or justifications for the reached conclusions. And, at last, they solve problems that generally fall into one of the following categories: interpretation, prediction, diagnosis, debugging, design, planning, monitoring, repair, instruction, and control. As a result of these distinctions, expert systems represent an

area of AI research that involves paradigms, tools, and system development strategies.

There exists already a wide variety of expert systems [1, 2, 3]. Instead of explaining all their details we rather develop on their general characteristics by pointing to the different domains of applications.

This paper is divided in 3 parts. In the first part (sections 2-4) we give the essentials of the knowledge representations schemes: logic, production systems, semantic networks. In the second part (sections 5-7) we explain the expert systems/shells: MYCIN/EMYCIN, HEARSAY-II/III and PROSPECTOR/KAS which are in the field of medicine, speech understanding and mineral exploration. We deliberately have chosen these widely differing fields in order to demonstrate the versatility of application, the difference in construction as well as the different realizations of the in section 2-4 introduced knowledge representation schemes. In section 8 we somewhat widen our outlook and aim at an overview about the possible applications of expert systems. In section 9 we present our conclusions.

2. LOGIC

In this section we present Logic which was one of the first representation schemes used in AI. Philosophers as Boole, Frege, Russel and others [4,5] made essential contributions to the development of this field. It has two important and interlocking branches. The first one focuses on 'what can be said' (relations, implications,.., axioms) and the second one on 'what can be derived'(rules of inference). Logic is a formal endeavor: it is concerned with the form, or syntax, of statements and with the determination of truth by syntactic manipulation of formulas.

The most fundamental notion in logic is that of truth. A properly formed statement, or "proposition", has one of two different possible truth values: TRUE or FALSE. Many of the things we say and think about can be represented in propositions that use sentential connectives to combine simple propositions:

AND($\wedge$,&), OR($\vee$), NOT($\neg$,$\sim$),
IMPLIES($\rightarrow$,$\supset$), EQUIVALENT($\leftrightarrow$, $\equiv$).

The use of the sentential connectives in the syntax of propositions brings us to the simplest logic, the "propositional calculus" in which we can construct compound propositions using the following truth table:

X	Y	X$\wedge$Y	X$\vee$Y	X$\rightarrow$Y	$\neg$X	X$\equiv$Y
T	T	T	T	T	F	T
T	F	F	T	F	F	F
F	T	F	T	T	T	F
F	F	F	F	T	T	T

We thus can build sentences of propositional logic just like expressions of mathematics. Among those we mention the tautology and the contradiction which are always TRUE resp. always FALSE.

In the propositional calculus we encounter the first rules of inference which allow for the deduction of a new sentence from previously given sentences. The power of logic lies in the fact that the sentence is assured to be true if the original sentences were true. The best known inference rule is "Modus Ponens". It states: if two sentences of the form X and X- Y are true, then we can infer that the sentence Y is true. Note that if we think of this as two entries in the database the modus ponens rule allows us to replace them with the simple statement Y.

We turn to the predicate calculus which allows for relationships between objects. It is an extension of the notions of the propositional calculus in that it allows for state-

ments about specific objects or individuals; these are called "predicates". A predicate is applied to a specific number of arguments -one, two,...several- and has a value of either TRUE or FALSE when "individuals", one might say constants, are used as the arguments.

We introduce two new notions, those of "variable" and "quantifier". A variable is a place holder, one that is to be filled in by some constant. There are two quantifiers, $\forall$, meaning "for all ...", and $\exists$, meaning "there exist ...". We can again have inference rules for quantifiers. The "elimination (or universal specialization) rule" states that, for any wellformed expression 0 with a variable X:

if we have $\forall X.\Phi(X)$ we can conclude that $\Phi(A)$.

There are two additional notions. "Functions", likes predicates, have a fixed number of arguments; but functions are different from predicates in that they do not just have the values TRUE or FALSE, but they "return" objects related to their arguments. The second important addition is the predicate: "EQUALS ( )". Two individuals X and Y are equal if and only if they are indistinguishable under all predicate functions. What we aim at with these additions is no longer pure predicate calculus; it is a variety of first-order logic.

The important feature of logic and related formal systems is that deductions are guaranteed correct to an extent that other representation schemes have not yet reached. The semantic entailment of a set of logic statements is completely specified by the rules of inference. Theoretically, the database can be kept logically consistent and all conclusions can be guaranteed correct. Other representation schemes are still striving for such a definition and guarantee of logical consistency.

One reason that logic-based representations have been so popular in AI research is that the derivation of new facts from old can be mechanized. Using automated versions of theorem proving techniques, programs have been written to determine automatically the validity of a new statement in a logic database by attempting to prove it from the existing statement.

The idea of using formal logic as a representation scheme and deductive inference as a reasoning method was apparently first suggested as an approach to commonsense reasoning and problem-solving by McCarthy in 1958 [6]. After several other initial attempts, Robinson arrived in 1965 at an automatic deduction technique for proving theorems which is relatively simple and logically complete [7]. Robinson's procedure and those derived from it are usually referred to as "resolution procedures" because the basic rule of inference they use is the "resolution principle":

From ($A \vee B$) and ($\neg A \vee C$) infer ($B \vee C$).

Robinson's work had a major influence on mathematical theorem proving, commonsense reasoning and problem solving.

First-order logic demands a clean syntax (form or grammer of the language), clear semantics (meaning of the formulas), and above all, the notions of truth and inference. Clarity about what is being expressed and about the consequences of, or possible inferences from a set of facts is perhaps the most important quality of this formalism. Logic is precise, flexible and modular. Its major disadvantage stems from the separation of representation and processing.

We point to a few logic-bases systems. The QA3-program by Green [8] was one of the earliest general-purpose questioning-answering systems that solved simple problems. The STRIPS-program (Stanford Research Institute Problem Solver) [9] was designed to solve planing problems faced by a robot in rearranging objects and navigationg in a cluttered environment. The representation scheme chosen for STRIPS was the first-order prodicate calculus. The FOL-system [10] is, among other things, a flexible proof checker for proving statements in first-order logic. Deduction is done with the natural deduction system of Prawitz [11].

Mechanical theorem proving goes back to the 1960's where Gilmore and Prawitz [12], shortly followed by Davis and Putnam [13], made the first attempts to implement Herbrand's procedure on digital computers. This AI branch has meanwhile considerably progressed and several useful computer systems, to a large extent based on Robinson's principle exist. We mention TPU by Chang and Lee [14], LMA/ITP by Wos, Overbeek, Lusk [15] and the BM-theorem prover by Boyer and Moore [16].

3. PRODUCTION SYSTEMS

In this section we consider the knowledge-representation by "production systems". This method was developed in 1972 by Newell and Simon [17] for models of human cognition.

The term "production system" is used to describe several different knowledge schemes, based on condition-action pairs, called "production-rules" (or P-rules). A production system consists of three parts: (a) a "rulebase" composed of a set of production rules, (b) a special, buffer-like "database" and (c) an "interpreter" which controls the system's activity. A production rule is a statement in the form: "IF this ‹condition› holds, THEN this ‹action› is appropriate". A typical rule might be:

```
IF    the patient has fewer, AND
      the patient has a running nose,
THEN  conclude that the patient has a cold.
```

During the execution of the production system, a P-rule, whose condition-part is satisfied, can fire; that is, it can have its action-part executed by the interpreter. Typical AI Systems nowadays contain hundreds of P-rules in their rulebase. The database is the focus of attention of the P-rules. The left-hand side of each P-rule in the rulebase represents a condition that must be present in the database before the P-rule can fire. The actions of the P-rules can change the database, so that other rules will have their condition part satisfied. The database may be a simple list, a very large array, or more typically, a medium-sized buffer with some internal structure of its own. The interpreter is a programm which has the special task of deciding which production to fire next.

Consider a production system that might be used to identify some items. The database (Db) consists of a list of symbols, and the rulebase and evaluation hints by the interpreter are as follows:

Rulebase:

```
P1 : IF ‹a›          THEN ‹b›
P2 : IF ‹c›          THEN ‹d›
P3 : IF ‹e OR b›     THEN ‹g›
P4 : IF ‹h AND i AND NOT(g)› THEN ‹j›
P5 : IF ‹g AND h›    THEN ‹k›
P6 : IF ‹h AND b›    THEN ‹l›
```

The condition-part of each of the P-rules corresponds to a question: Is the item ‹a› TRUE, meaning is it in the database, and similarly is the item ‹c› satisfied?, and so on. The action-parts of the P-rules represent additions to our knowledge about the unknow item.

Interpreter:

Step 1 : Find all P-rules whose condition parts are TRUE and make them applicable.

Step 2 : If more than one P-rule is applicable, then deactivate any P-rule whose action adds a duplicate symbol to the database.

Step 3 : Execute the action of the lowest numbered (or only) applicable P-rule. If no P-rules are applicable, then quit.

Step 4 : Reset the applicability of all P-rules and return to Step 1.

Production systems operate in cycles. In each cycle, the P-rules are examined in a manner specified by the interpreter to see which are appropriate and could fire. Then, if more than one is found appropriate, a single P-rule is selected from among them. Finally, the P-rule is fired. These three phases of each cycle are called "matching", "conflict resolution" and "action".

Suppose the database originally consists of the list:

$$Db = \{ a, h \} .$$

The first cycle starts with Step 1 of the interpreter algorithm: Since only P1 is applicable, Step 2 is not necessary, and Step 3 causes the action part of P1 to be executed. This adds the symbol ‹b› to the Db-list, representing a new fact about the unknown item:

$$Db = \{ b, a, h \} .$$

Step 4 ends the first cycle and brings us back to Step 1 - finding all the applicable P-rules.

In the second cycle the (P1, P3, P6)-rules are applicable. So, in Step 2 we must check if any of these three adds a duplicate symbol to the database. P1 adds ‹b›, which is a dublicate, so it is eliminated. Then in Step 3 we select P3 to be executed (because it has a lower number than P6), resulting in:

$$Db = \{ g, b, a, h \} .$$

In the third cycle we find that the (P1, P3, P5, P6)-rules are applicable. Checking, in Step 2, for redundant entries, we eliminate the (P1, P3)-rules from consideration. In Step 3, we decide to execute P5, once again because it comes before P6. This results in the database:

$$Db = \{ k, g, b, a, h \} .$$

In the next two cycles of execution, our sample production system will finish. In cycle 4, the symbol ‹1› is added to the database, and in the last cycle, finding no non-redundant P-rules to fire, the interpreter finally quits, leaving the database-list:

$$Db = \{ 1, k, g, b, a, h \} .$$

The indicated items and in particular the last one can in this way be deduced from the initial Db-list and the P-rules (see Fig. 3.1).

The research on deductive inference has recognized two fundamentally different ways that people reason. Sometimes we work in a data driven, event driven or bottom-up direction, starting from the available information as it comes in and trying to draw conclusions that are appropriate to our goals. This is how our sample production system worked, for example. In production-system research this is called forward chaining method of inference (Fig. 3.1). We sometimes work the otherway, however, starting from a goal or expectation of what is to happen and working backwards, looking for evidence that supports or contradicts our hunch. This is called goaldriven, expectation-driven or top-down thinking, and in production systems it is referred as backward chaining methode of inference (Fig. 3.1). It requires looking at the action parts of rules to find ones that would conclude the current goal, then looking at the left-hand sides of those rules to find out what conditions would make them fire, then finding other-rules whose action parts conclude these conditions and so on.

One obvious quality of production systems is modularity which means that the individual P-rules in the rulebase can be added, deleted or changed independently. They behave much like independent pieces of knowledge, and communicate only by means of the data in the database. A uniform structure is imposed on the knowledge in the rulebase and all information is encoded with the rigid structure of production rules. The structure of the left and right-hand side of the P-rules has been progressively extended. The left-hand side can now evaluate an arbitrarily complex condition, and the form of the right-hand side also includes variables.

Production systems have been used to represent real world tasks, like speech understanding, medical diagnosis, or mineral exploration and in psychology, they were used for modelling human behaviors. The following utility criteria were proposed:

1. Domains in which the knowledge is diffuse, consisting of many facts (medicine), as opposed to domains in which there is a concise, unified theory (physics).
2. Domains in which processes can be represented as a set of independent actions, as opposed to domains with dependent subprocesses.
3. Domains in which knowledge can be easily separated from the manner in which it is to be used, as opposed to cases in which representation and control are merged.

The above-presented knowledge-representation scheme has been used, most typically, in the MYCIN-System [18] which acts as a medical consultant for the diagnoses and therapy of bacteremia and meningitis infections. It also includes a knowledge acquisition subsystem, THEIRESIAS [19], which helps

expert physicians expand or modify the rulebase. MYCIN's rulebase contains several hundred production rules representing human-expert level about the domain. The system is distinguished by its use of a backward chaining control structure and inexact reasoning involving confidence factors that are attached to the conclusion part of each rule to help determine the relative strength of alternative diagnoses. There exist a number of similar expert-systems such as PUFF [22], EXPERT [20], AM [21] ..... and many others which might differ in their construction details but which use the same underlying knowledge-representation scheme.

4. Semantic Networks

In this section we consider the knowledge representation by "Semantic Networks". This method was developed in 1968 by Quilian [23] for psychological modelling and by Raphael [24] for the question-answering system SIR.

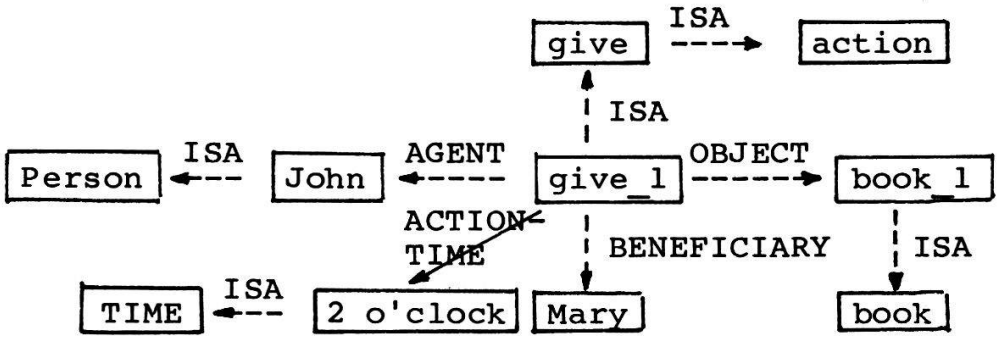
The term "semantic networks" stands for a class of representation formalisms that share a common notation consisting of nodes and arcs; it is a scheme for representing abstract relations among objects (in a knowledge base), such as membership in a class: Robin ISA Bird, Sparrow ISA Bird, Bird ISA Animal. Such relationships may be represented graphically by a network of nodes and arcs where the nodes represent objects (e.g. Sparrow, Bird, Animal) and the links represent the relations (e.g. ISA) among the objects. Semantic networks were originally designed as a way to represent the meanings of words. Suppose we wish to represent a simple fact like: "All Robins are Birds", in a semantic network.

We might do this by creating two nodes to designate: "Robins" and "Birds" with an arc connecting them. If "Clyde" were a particular individual who we wished to assert is a robin, we could add a node for "Clyde". Please ignore the WINGS-part for the moment. In this example we have represented not only the two initial facts, but we also deduced a third fact, namely, that "Clyde" is a "Bird", simply by following the ISA-links. The ease with which it is possible to make deductions about inheritance hierarchies, is one reason for the popularity of the semantic networks as a knowledge representation.

One often needs to represent knowledge about properties of objects. For example one might wish to express the fact "Birds have wings", which is expressed by the WINGS-part on the right-hand side in the above chart. Our representation makes it easy to deduce that "Robin" and "Clyde" also have wings. All that is necessary is to trace up the ISA-hierarchy, assuming any facts asserted about higher nodes on the hierarchy can be similarly considered valid for the lower ones. Note that each property stores a one-way arc. To store bidirectional links, it is necessary to store each half separately, possibly with a different label. Semantic nets are usually represented using some kind of attribute-value memory structure. So, for example, in LISP, each node would be an atom, the arcs would be properties, and the nodes at the other ends of the links would be the values:

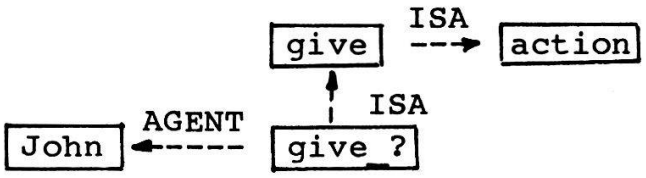
ATOM	PROPERTY LIST
CLYDE	((ISA ROBIN))
ROBIN	((ISA BIRD))
BIRD	((HAS-PART WINGS))

From the discussion sofar, it is clear that semantic nets can be used to represent relationships that would appear as two-place predicates in predicate logic. Many one-place predicates can be thought of as two-place predicates using some very general purpose predicates such as ISA. Three or more place predicates can also be converted to a binary form by creating one new object representing the entire predicate statement and then introducing binary predicates to describe the relationship to this new object of each of the original arguments. The sentence: "John gave the book to Mary" is in predicate logic expressed by: gave (John, Mary, book).
The analogous semantic net is shown in the following graph:



In the above example give_1 is the newly introduced object. It is considered as a particular member, or instance of the class give since it is correlated to John, Mary,.. . Similarly book_1 is an instance of the general class book. Obviously, the above example now admits additional information such as for instance on time 2 o'clock, and other particularities.

The reasoning mechanism used by many semantic network systems is based on the "matching network structures". A network fragment is constructed, representing a sought-for object or a query, and then matched against the network database to see if such an object exists. Variable-nodes in the fragment are bound in the matching process to the variables they must have to make the match perfect. For example, suppose we use the above sketched semantic network for the sentence: "John gives Mary the book" and we wish to answer the question: "What did John?". We might construct the fragment:



which represents an instance of an action in which John is the agent. This fragment is then matched against the network database looking for all nodes where John is the AGENT. In our case the only node is give_1 and the answer would be:

"John gives". Had no match been found, the answer would have been: "John didn't do anything".

Semantic networks are a very popular knowledge representation scheme. The node-and-arc structure captures something essential about symbols and pointers in symbolic computation and about association in the psychology of memory. Most current work involves elaboration of the semantic net idea, in particular, work on aggregate network structures called "frames".

One of the major problems is how to handle quantification. One way of solving the problem is to partition the semantic network into a hierarchy of spaces, each of which corresponds to the scope of one or more variables.

Besides computational problems that arise when network databases become large enough to represent non-trivial amounts of knowledge, there are many, more subtle problems involving the semantics of the network structures.

The node-and-link formalism of semantic netwoks has found use in a number of AI systems in different application domains. We point to the early spreading activation model by Quilian [23] and to Raphael's [24] questioning-aswering system SIR.

A good example of a network deduction system, constructed around the matching paradigm, is SNIFFER [25]. It has the general power of a theorem prover for making deductions from the network database and is capable of taking advantage of heuristic knowledge embedded in procedures providing advice about which network elements should be matched first and about how to match the selected elements.

Semantic networks were also used in expert-systems. PROSPECTOR [26], for example, is a computer-based consultation system to assist geologists working in mineral exploration. It's mainfunction to match data from a particular situation against models that are descriptions of the most important types of arc-deposits. The data are primarly surface geological observations - uncertain and incomplete - so that the conclusion is expressed as a probability or a degree of match.

5. MYCIN

The MYCIN [27] system (see Fig. 5.1) addresses the problem of diagnosing and treating infectous blood diseases such as bacteremia and meningitis. Its knowledge comprises of approximately 450 rules relating possible "premises" to associated "actions". In its problem-solving, MYCIN tests a rule's conditions against available data or requests data from the physician. If appropiate, it tries to infer the truth or falsity of a condition from other rules.

5.1 MYCIN System

The knowledge in MYCIN is encoded in "production-rules" with an example given in Fig. 5.2. The rules are stored internally in LISP-form from which the English version is generated. Each rule is a single "chunk" of domain-specific information indicating an "action" that is justified if the conditions, specified in its "premises", are fulfilled. Since the rule employs a vocabulary of concepts common to the domain, its form itself involves a comprehensible part of domain knowledge. Each rule is highly stylized, with an IF...THEN format and a specific set of admissible "primitives". In fact, the internal form is executable LISP code. This highly structured form makes it possible for a program to be designed to examine the rules as well as to execute them. For example, the rules can be translated into readable English format, as shown above. The translation capability has been used in MYCIN to explain the program's inferences to the expert.

The "premise" of each rule is a Boolean combination of one or more "clauses", each of which is constucted from a "predicate function" with an "associated triplet": ‹object, attribute, value› as its arguments. Thus, each "premise-clause" typically has the following four components: ‹predicate function› ‹object› ‹attribute› ‹value›. For example, the second clause in the above Rule-050: ‹‹the site of the culture is one of the sterile sites››, reads

```
(MEMBF       CNTXT      SITE       STERILESITES)
   |           |          |            |
‹Predicate› ‹Object› ‹Attribute› ‹Value›
```

MEMBF is a predicate, and the triple says that the site of the current ‹object› (an organism, implicitly referred to by CNTXT) is a member of the class of sterile sites. In general, the content of an "associate triplet" can be translated into English as follows:

THE ‹attribute› OF ‹object› IS ‹value›

A standardized set of

‹predicate functions›: MEMBF, SAME, KNOWN, BELONGING-TO, IS, (NON-)SUSPECTED, DEFINITE,..

‹objects›	: ORGANISM, CULTURE, MEDICAMENT, PATIENT, INFECTION,...
‹attributes›	: SITE, PORTAL, LOCUS, GRAM, IDENTITY, SENSITIVS, CONTAMINENTS,...
‹associated values›	: E.COLI, KLEBSIELLA, PENICILLIN, BLOOD,...

form the vocabulary of the "conceptual primitives" for constructing rules.

The ‹objects› (called "contexts" in MYCIN) in the associated triplets are variables corresponding to domain entities. These entities are instantiated and organized by the program into a simple hierarchy called the "context-tree". For example, in the MYCIN domain the ‹objects› might be PATIENT-1, CULTURE-1, ORGANISM-1, and ORGANISM-2, and the "context-tree" would indicate that ORGANISMs belong to CULTUREs and CULTUREs belong to PATIENTs. The "context-tree" provides some of the inheritance mechanisms of a frame representation. For example, since cultures also have sites, the system can discover the site of ORGANISM-2 by knowning the ORGANISM-2 came from CULTURE-1 and looking up the site of CULTURE-1. The ‹attribute› (called "clinical parameter" in MYCIN) represents the characteristic of an ‹object› in the "context-tree": the name of a patient, the site of a culture,... . The ‹attributes› known to MYCIN are therefore categorized in accordance with the particular ‹object› to which they apply. Each of the 65 ‹attributes› currently known to MYCIN has an associated set of properties that is used during its consideration for a given ‹object›. The ‹value› of every ‹attribute› in the associated triplet is stored by MYCIN along with an associated certainty-factor that reflects the system's "belief" that the value is correct.

A "rule-premise" is always a conjunction of "clauses", but it may contain arbitrarily complex conjunctions or disjunctions nested within each clause. Instead of writing rules whose "premise" would be a disjunction of clauses, a separate rule is written for each clause. The action-part indicates one or more conclusions that can be drawn if the premises are satisfied, making the rules purely inferential.

To summarize, there are two major forms of knowledge representation: (i) the ‹objects, attributes, values› which form a vocabulary of domain specific conceptual primitives, and (ii) the inference rules expressed in terms of these primitives.

Note that the rules are judgmental, that is, they make inexact inferences. To accomodate uncertainty, MYCIN associates a "certainty-factor (CF)" with every ‹object, attribute, value› triple. This number, a normalized probability, ranges from -1 (when the triple represents a false assertion) through 0 (no opinion) to +1 (unquestionably true). Medical facts about the patient are thus represented as 4-tuples made-up of the "associative triple" and its current CF. Positive CF's indicate that the evidence confirms the hypothesis;

negative CFs indicate disconfirming evidence. The following are examples of such 4-tuples:

‹object›	‹attribute›	‹value›	CF
(ORGANISM-2	IDENT	KLEBSIELLA	+0.2)
(ORGANISM-2	IDENT	E. COLI	+0.7)
(ORGANISM-1	SENSITIVS	PENICILLIN	-1.0)
(PATIENT-1	IMMUNOSUPRESSED	YES	+1.0)

The ‹predicates›, according to their particular nature and definition, evaluate, for some certainty interval such as 0.2 to 1.0, to the CF of their argument-triple, or they can take the values TRUE or NIL in specific situations; they even can be fuzzy-set functions that indicate a degree of truth. The logical connective AND returns the minimum, and analogously the connective OR returns the maximum value of the CFs of its arguments. As a condition of applying a rule, a "premise" is considered to be TRUE if its certainty is greater than some threshold, typically +0.2, and FALSE if its certainty is less than another threshold, typically -0.2.

The second use of CF's is in the statement of the "production-rules" themselves. For example in the above Rule-050, the asserted conclusion is weighted with a mild degree of confidence: 0.7. This CF is thus a measure of the association between the "premise-clauses" and the "action-clauses" of each rule. The term "TALLY" in the LISP-form of the "rule-action" indicates MYCIN's believe in the "rule-premise".

A "fired" rule updates the CFs of the specified "action(s)", or, as an alternative, it evaluates a set of attached procedures. In doing this, the system combines: (i) the CF of the "rule-premise", (ii) the present CF of the "rule-action", and (iii) the CF associated with the rule. The CF of a new fact, derived from a rule, is given by:

$$CF = CR \times \min[\ldots CFi \ldots]$$

The CR is the certainty-factor attached to the rule and the CFi is the certainty-factor of the i-th "premise-clause". Thus if the premise was only weakly believed (low, positive CF), any conclusions that the rule might make would be modified (reduced) to reflect this weak believe. If two different rules lead to the same conclusion with the certainty-factors CR1 and CR2, they enforce each other:

$$CF(\text{conclusion}) = CR1 + CR2 * (1 - CR1)$$

These simple rules are in fact based on a slightly more complicated model of inexact reasoning [28] which, to a substantial part, can be traced back to probability theory with the

assumption of statistical independence [29]. The alternative action of evaluating the attached procedures, is an escape mechanism that allows the execution of arbitrary LISP-code.

MYCIN's model of inexact reasoning permits the coexistence of several plausible values for a single ‹attribute› if this is suggested by the evidence. For example, after attempting to deduce (IDENT) of an organism, MYCIN may have concluded correctly that there is evidence of both E. COLI and KLEBSIELLA.

The mechanism to draw conclusions from the rules in the knowledgebase and the current data, is called "inference engine". In MYCIN, rules are invoked in a goal-directed backward-chaining fashion that results in an exhaustive "depth-first search" of an AND/OR goal tree with the sougth topic at its top. Backward-chaining can go several levels deep; for a "rule-action" to be true, the "premise-clauses" must be true. However, each of them might itself be the "action-part" of another rule, and so on.

For example, assume that the program is attempting to determine the identity of a particular infection organism. It thus retrieves all the rules that make a conclusion about that topic and invokes each one in turn. The above Rule-050 for instance mentions in its "action-part": IDENT (organism) = bacteria. In order to certify the validity of the rule, each of its "premise-clauses" must be evaluated to see whether the rule's conditions can be satisfied. This process begins with the first "premise-clause", where: INFECT = primary bacteremia. Since the type of the infection is not given by the data and it therefore is not known, the system sets up a new subgoal and the process recurs. The system now looks for rules that conclude about this new topic: the type of the infection?, leading possibly to several answers with differnt degrees of certainty. Among these answers one might find the first "premise-clause" being certified.

We give another example in Fig. 5.3. Note, the subgoal that is set up is a generalized form of the original goal. It is always of the form: Determine the value of the ‹attribute›, rather than: Determine whether the ‹attribute› = ‹value›.
Thus, for the first "premise-clause" in Rule 050, the subgoal is: Determine the type of infection. By setting up the generalized goal of collecting all evidence about an ‹attribute›, the perfomance program effectively exhausts each subject as it is encountered and thus tends to group together all questions about a given topic. This feature results in a system that displays a much more focused, methodical approach to the task, which is a distinct advantage when human-engineering considerations are important. Obviously, this leads to the deduction and collection of information that is not strictly necessary. However, since such unnecessary efforts occur rarely - only when the ‹attribute› can be deduced with certainty to be the ‹value› named in the original goal - is has not proven to be a problem in practice.

The search is thus "depth-first" because each "premise-clause" is thoroughly explored in turn. The resulting search is an AND/OR goal tree because the "premise-clauses" also may have OR connections. The search is exhaustive because all applicable rules are "fired" and their conclusions are rank-ordered by the certainty-factor. Since the rules are inexact they lead to conclusions of less than total certainty. Thus even if one rule succeeds, the system continues to collect further evidence about the subgoal from other applicable rules; MYCIN considers all possibilities.

We give a toy example. Suppose MYCIN's goal is to find the value of A, and some of its rules are

R1 : IF ‹F=f › THEN ‹C=c1, 0.5›
R2 : IF ‹G=g AND H=h › THEN ‹C=c1, 0.6›
R3 : IF ‹H=h AND I=i › THEN ‹C=c2, 0.7›
R4 : IF ‹B=b AND C=c1› THEN ‹A=a, 0.8›
.. and other rules making conclusions about A ..

Suppose MYCIN also knows that the values of B, F, G, H, I, E are laboratory data, determined by asking the user for their values. The AND/OR tree corresponding to these rules is shown in Fig. 5.4. MYCIN searches this graph depth-first from left to right, determining the values of B, F, G, H, I, C and A in turn. Note that when a rule such as R4 is involved, the subgoals MYCIN creates are not to prove that B=b and C=c, but rather to find the values of B and C. The system can then focus on a particular topic when interacting with the user, rather than jumping from topic to topic. In addition B and C need not be reevaluated if another rule is ever encounted that requests information about them.

If, after trying all relevant rules to resolve a subgoal, the total weight of the evidence about a hypothesis falls between -0.2 and +0.2 (an empirically determined threshold value), the answer is regarded as still unknown. This result would occur: if no rules were applicable because their premises did not match the available data, if the applicable rules were too weak, if the effects of several rules offset each other, or if there were no rules for this subgoal at all. In any of these cases, when the system is unable to deduce the answer, it asks the user for the value of the subgoal.

This strategy, of always attempting to deduce the ‹value(s)› of a subgoal and asking the user only when deduction fails, ensures a minimum of questions. This, however, can lead to an extended search for a subgoal with a less than definite answer, although the answer is known with certainty. Some of the ‹attributes› have therefore been labeled "laboratory data" to indicate that they represent definite information from quantitative tests. In these cases the system attempts to deduce the answer only if the user cannot supply it.

Two other additions increase the inference engine's

efficiency. First, before the entire list of rules of a subgoal is retrieved, the program looks for a sequence of rules that would establish its "action-part" with certainty, based on what is currently known. Since this is a search for a sequence of (mainly definitional) rules with CF=1, the result is termed a "unity path". In addition to efficiency, this process offers the advantage of allowing the program to make commonsense deductions with a minimum of effort. Second, the inference engine performs a partial evaluation of the "rule-premises". The value of one or several "premise-clause" may already have been established while the rest is still unknown. If this clause alone would make the premise false, there is clearly no reason to do all the search necessary to establish the others. Each "premise" is thus previewed by evaluating it on the basis of the currently available information which gives immediate insight whether the rule is guaranteed to fail.

The "meta-rules" are implementations which allow the system to guide its search process. They are strategic rules that prevent an exhaustive enumeration by indicating the best approach to determine a subgoal.

One of the meta-rule concerns, for instance, the general aim to provide a therapy for a patient; it reads:

```
   IF  ‹a therapy is wanted›
 THEN  ‹consider, in the given order, the rules for:
     1) acquiring clinical information about the patient,
     2) finding which organisms are the infection-cause,
     3) identifying the most probable organisms,
     4) finding all the potentially useful medicaments,
     5) choosing a small number of most adepted rules ›
```

The meta-rule 001 of the MYCIN system, as another example, has the form:

```
 IF ‹the culture was not obtained from a sterile source› AND
    ‹there are rules which mention in their premise a
     previous organism which may be the same as the
     current organism›
THEN ‹it is definite (1.0) that each of them is not
      going to be useful›
```

with its content encoded in LISP-form:

```
   (AND  (NOTSAME  CNTXT  SITE  STERILESOURCE)
         (THEREARE OBJRULES)
         (MENTION CNTXT PREMISE SAMEBURG) SET1))
   (CONCLIST SET1 UTILITY NO TALLY +1.0)
```

It is important to note the character of the information conveyed by the meta-rules. First, note that in all cases we have a rule that is making a conclusion about other rules. That is, where "production-rules" conclude about the medical do-

main, meta-rules conclude about the "production-rules". They can make deductions about the likely utility of "production-rules", or they can indicate a partial ordering between two subsets of "production-rules". Note also that meta-rules make conclusions about the utility of "production-rules", not about their validity. This is important because it has an impact on the question of distribution of knowledge.

Adding meta-rules to the system requires only a minor addition to MYCIN's control structure. As before, the system retrieves the entire list of rules to the current goal. But before attempting to invoke them, it first determines if there are any meta-rules relevant to the goal. If so, these are invoked first. As a result of their actions, we may obtain a number of conclusions about the likely utility and relative ordering of the rules-list. The conclusions are used to reorder or shorten the rules-list, and the revised list of rules is then used. Viewed in tree-search terms, the current implementations of meta-rules can either prune the search space or reorder the branches of the tree.

MYCIN's explanation program allows a user to examine both the reasons for the conclusions reached in a particular session and the information in the static data. This can be done either through the use of simple "WHY" and "HOW" commands when the system requests the ‹value› of an ‹attribute› or through the keyword parser that can interprete simple requests given in English.

The representation of knowledge as "production-rules" and the ability to explain specific rules allow MYCIN to interact with an expert clinic in a manner that permits the system to acquire new knowledge. The THEIRESIAS system [30] is a high-level knowledge base editor that works in conjunction with MYCIN and assists in entering and updating the (large) MYCIN knowledge base, finding errors in the database, modifying faulty rules, or adding new rules. It checks the rules for syntactic validity, sees that they do not contradict or subsume existing rules and inspects faulty reasoning chains. The THEIRESIAS' rule acquisition process is based on a record of MYCIN's search. Rule acquisition is guided by a set of rule models that dictate the form and indicate the likely content of new rules. Rule models are not given in advance, but are inferred from the knowledge base of existing rules.

5.2 EMYCIN

The EMYCIN system [31] is basically a domain-independent or an appropriate skeletal version of MYCIN that can provide consultative advice. The basic control strategy employed by EMYCIN is backward-chaining, its initial goal being to determine the value of a top-level attribute. At any subsequent time, EMYCIN is working on the goal of establishing the value of the attribute of some object. To do this, it retrieves a precomputed list of rules whose consequents are known to bear on that goal, and it systematically attempts to apply the ru-

les until it either establishes the value with complete certainty or exhausts the rule list. If no value can be deduced - whether because there are no rules or because the rules were unsuccessful - it resorts asking the user for the value.

The resulting consultation system takes as input a body of measurements or other information pertinent to a case and produces as output some form of recommendation or analysis of the case. This framework seems well suited for many diagnostic or analytic problems, notably some classes of fault diagnosis, where several input measurements (symptoms, laboratory tests) are available and the solution space of possible diagnoses can be enumerated. It is less suited for "formation" problems, where the task is to piece together existing structures according to specified constraints to generate a solution.

EMYCIN is not designed to be a general-purpose representation language. It is thus wholly unsuited for some problems. The limitations derived largely from the fact that EMYCIN has chosen one basic, readily understood representation for the knowledge in a domain: production rules that are applied by a backward-chaining control structure and that operate on data in the form of associative triples. The representation as implemented in EMYCIN, is unsuitable for problems of constraint satisfaction, or those requiring iterative techniques. Among other classes of problems that EMYCIN does not attempt to handle are simulation tasks and tasks involving planning with stepwise refinement. One useful heuristic in thinking about the suitability of EMYCIN for a problem is that the consultation system should work with a "snapshot" of information about a case. Good advice should not depend on analyzing a continued stream of data over a time interval.

5.3 MYCIN Family

MYCIN is the result of a concentrated effort within the Stanford Heuristic Programming Project to use "production-rules" as a knowledge representation. During its development and particularly after its successful completion several related projects were persued which are shown in Fig. 5.5.

DENDRAL [32] was the forerunner of MYCIN in the sense that many of the lessons learned in its construction were used in the design and implementation of MYCIN. GUIDON [33] is a program for teaching MYCIN's infectious-disease rules to students. Its teaching knowledge is stated in the form of 200 tutorial rules which include methods for guiding the dialogue economically, presenting diagnostic rules, constructing a student model, and responding to the student's initiative. Rather than teaching a student rule clause by rote, it is advantageous to convey an approach strategy for bringing those steps to mind. To make this implicit design knowledge explicit the NEOMYCIN system [34] is being developed.

EMYCIN [31] has been used as the starting point for a set of application oriented expert systems of similar structure.

We mention: SACON [35] is a system for advising structural engineers in the use of a large, finite element analysis program for modelling various mechanical structures. ONCOCIN [36] assists physicians with the managment of patients enrolled in experimental plans for treating cancer with chemotherapy. PUFF [37] /CENTAUR [38] /WHEEZE [39] diagnose patients with pulmonary (lung) function disease. CENTAUR is based on hypothesis-directed reasoning, and WHEEZE provides for a uniform declarative representation of the domain knowledge. VM [40] monitors the post-surgical progress of patients (after cardiac surgery) requiring a device called "mechanical ventilator" which provides breathing assistance, with the type and setting of the ventilator being adjusted to match the patients need. GRAVIDA [41] reasons about complications of pregnancy, CLOT [42] about blood clotting disorder, and HEADMED [43] gives diagnoses and treatment for psychiatric patients. Other recent MYCIN-based systems are: DART [44], LITHO [45], BLUEBOX [46], ...

6. HEARSAY

The HEARSAY-II speech understanding system [47] has been one of the most influential AI-programs, not so much because of its speech unterstanding ability, but in the way it is constructed; there are several "knowledge sources (KS)" cooperatively solving a problem by posting hypotheses on a global "blackboard". This modular architecture - the KSs don't address each other directly - proved to allow for great flexibility in combining different knowledge sources and investigating various control strategies in a system. In problem domains characterized by a large search space, by the need to combine different kinds of knowledge, and by ambiguous or noisy data, HEARSAY's modular architecture has proved especially well suited.

6.1 HEARSAY-II System

The system architecture of HEARSAY-II is shown in Fig.6.1. We notice on the right-hand side the KSs (or "Knowledge Sources") and on the left-hand side the "blackboad"; we further point to the "blackboard monitor" and the "scheduler". The encircled units are "programm modules" and those in frames indicate "databases". The solid lines (or arcs) indicate the data-flow and the dashed lines the control-flow.

The "blackboard" is the systems global database. It is divided into a number of levels corresponding to a hierarchical breakdown of the speech analysis. The different levels are indicated on the left-hand side of Fig.6.2. The "blackboard" is thus subdivided into a set of information levels corresponding to the intermediate representation levels of the decoding process (wave form of utterance, sound segments, syllable classes, words, word sequences, phrases); they can be thought of as the various levels in a "problem-reduction tree" at which sub-problems are located. The sequence of levels on the "blackboard" forms a loose hierarchical structure: hypotheses at each level aggregate or abstract elements at the adjacent lower level. The possible hypotheses at a level form a search space for KSs operating at that level. A partial interpretation at one level can constraint the search at another level. The hypotheses on the "blackboard" are arranged along two dimensions: level and time. The time dimension takes account of the time periods of the utterance being analyzed. The goal of the system is to create a single hypothesis that represents a solution to the problem: an acceptable interpretation of an entire utterance.

The "knowledge sources (KSs)" are pattern-invoked programs, meaning they encode domain-dependent knowledge in the form of operators, such as for instance "production-rules". They have the role of generating, combining and evaluating hypothetical interpretations and are therefore diverse and independent. The necessity for diverse KSs derives from the diversity of tranformations needed to arrive at an interpretation of an acoustic signal. Each KS can be

schematized as a conditions-action pair. The "condition component (or program)" prescribes the situations in which the KS may contribute to the problem-solving activity, and the "action-component" specifies what that contribution is and how to integrate it into the current situation. Each KS looks at the hypotheses posted on one level, called its "structure frame", and in turn posts its hypotheses on another (possibly the same), or on several levels, called the "response frame". The KSs have been developed to perform a variety of functions. These include extracting acoustic parameters, classifying acoustic segments into phonetic classes, recognizing words, parsing phrases, and so on. For example, the KS (PREDICT) works completely within the phrase level on the "blackboard", predicting the word that might extend a phrase. In contrast the KS (VERIFY) looks for acoustic evidence in the signal for hypotheses at the word level. The KSs indicated by the transition arcs between the levels in Fig. 6.2 are those of the 1976 C2-configuration of the HEARSAY-II system [47]. The processing at the lower-part of the "blackboard" was accomplished by the KS for acoustic segmentation (SEG) and word-spotting (POW, MOW, WORD-CTL). The KS (SEG) abstracts a string of allophones from the acoustic signal. These are assigned to syllable classes by the KS (POM), and the syllable classes are used by the KS (MOW) to hypothesize word. Note that HEARSAY's lexicon is organized by syllable classes; each section of the lexicon contains pronounciations of all the words that make one syllable-class. The creation or modification of a hypothesis at any level immediately invokes the KS (RPOL) which rates the credibility of the hypotheses. The number of hypotheses that the KS (MOW) can make is controlled by the KS (WORD-CTL). The WIZARD procedure [48] scores the hypothesized words by comparing their acoustic characteristics to the stored representations of word-pronounciations. The processing at the upper-part of the "blackboard" involves predicting, testing and concatenating multiple-word sequences, one or more of which will eventually account for all of the words spoken. The KSs (WORD-SEQ) and (Word-SEQ-CTL) extend those words recognized by the lower-part of the "blackboard" into a small number of islands of one or more words, using a data structure that contains all legal pairings of words. By hypothesizing extensions of the newly hypothesized words, the islands can be extended recursively. However, the syntax of the islands was generated from legal pairs of words and therefore the longer islands may not be syntactically correct. The KS (PARSE) checks the syntax. When a number of multiple-word islands are developed, the KS (VERIFY) tries to check each word against the segmented acoustic signal in the context of its island. The KSs (PREDICT) and (CONCAT) are also used to extend hypothesized word sequences. The KS (PREDICT) generates all the words that can immediately precede or follow a word wequence, while the KS (CONCAT) tries to join word sequences together to form longer ones. Finally, the KS (STOP) is used to terminate proces-

sing of the speech signal, either because the best interpretation of the sentence has been found or because too much processing time has been used. The KS (SEMANT) generates machine instructions to carry out the spoken command. The system contains approximately 40 KSs, which are from 5 to 100 pages of source code a piece. Thirty pages is a typical KS size. Each KS has up to 50 Kbytes of its own local data storage.

The "processing" in HEARSAY-II combines both top-down and bottom-up. The first type, associated with means-ends analysis and problem reduction strategies [49] attempts to reach a goal by dividing it into a set of simpler subgoals and reduceing these recursively until only primitive or immediately solvable subgoals remain. Examples of top-down processing include the reduction of a general sentential concept into alternative sentence forms, each sentence form into specific alternative word sequences, specific words into alternative phone sequences and so on, until a last interpretation is identified. The second, or bottom-up, method attempts to synthesize interpretations directly from characteristics of the data provided. One type of bottom-up method would employ procedures to classify acoustic segments within phonetic categories by comparing their observed parameters with the ideal parameter values of each phonetic category. Other bottom-up procedures might generate syllable or word hypotheses directly from sequences of phone hypotheses, or might combine temporally adjacent word hypotheses into syntactic or conceptual units. As a result, processing at the lower-part of the "blackboard" is strictly bottom-up. The KSs (SEG, POM, MOW) are activated in that order, and the processing done by one is completed before another is activated.

The "scheduler", a special KS endowed with knowledge about how to conduct best the search in a particular domain, determines the KSs to be activated next. It adapts automatically to changing conditions of uncertainty by changing the breadth of search, using as a basic mechanism the interaction between KS-assigned credibility ratings on hypotheses, and scheduler-assigned priorities of pending KS activations. Messages posted on the "blackboard" are noted by the "blackboard monitor", which creates entries on the scheduling queues for any KS whose applicability condition might be satisfied. When a KS is activated, it examines the current contents of the "blackboard" and applies its knowledge either to create a new hypothesis and write it on the "blackboard", or to modify an existing one. Although the execution of the entire HEARSAY-II system consists of the asynchronous execution of a collection of KSs, the execution of an individual KS is a sequential process. Once a KS is activated, it executes without being interrupted until it is finished.

We comment on the design ideas of the HEARSAY-II system, with its architecture being summarized by: (i) separate, anonymous knowledge sources, (ii) self-activating, asynchronous, parallel procesing, (iii) globally accessed, structured data-

base, (iv) goal-directed problem reduction, and data-directed knowledge invocation. The main features that distinguish the HEARSAY-II architecture from that of systems such as MYCIN are the use of arbitrary pattern-invoked programs as units of knowledge rather than "production-rules" and the flexibility of the "scheduler" as opposed to the strict goal-driven invocation used in MYCIN. For a large, complex problem such as speech understanding, these features offer several advantages. Since the KSs can be arbitrarily complex - and arbitrarily different in their internal operation - the most appropiate problem solving approach can be implemented at each level of processing. Each KS may itself be a small knowledge-based problem-solver, and its internal processes have only local effects, rather than causing potential interactions with the rest of the system. The multiple levels of the database provide the necessary abstractions for searching a large solution-space, and the levels are heterogeneous to match the diversity of the interpretation knowledge. The opportunistic scheduling combines the least-commitment idea with the ability to manage computational resources by varying the breadth of search and by combining top-down to bottom-up processing. These qualities alleviate the "combinatorial explosion" that often occurs when search techniques are used on very large problems. In fact, when portions of HEARSAY-II were experimentally rewritten as a "production system", the system ran approximately 100 times slower.

6.2 HEARSAY-III

The techniques developed in HEARSAY-II have since been generalized into the HEARSY-III system [50] a domain independent framework for building large knowledge based expert-systems. Its overall design concept was based on a set of requirements abstracted from HEARSAY-II. Deemed particularly important were facilities to do the following:

- support codification of diverse, general sources of knowledge
- support their application and flexible coordination,
- represent and manipulate competing solutions that are constructed incrementally
- reason about partial solutions
- describe and apply domain-dependent consistency constraints to the competing partial solutions
- support long-term, large-system development and experimentation.

HEARSAY-III is built on the AP3-language [51], implemented in INTERLISP [52], which offers a data base structure similar to the PLANNER-like languages [53]. Atomic facts are held in a "declarative" data base which also may hold some deductive rules. The inference engine does pattern matching for retrieval and rule application and controls the process of deduction and transformation of the data base. Facts in the data base are ordinarily used representationally in much the same way as the formulas of logic. The knowledge source triggers,

the inference rules and constraints, and the context mechanism, allowing for reasoning along independent patterns, are thus expressed in AP3.

In HEARSAY-III, the time dimension of the "blackboard" has been removed, since it is not appropiate to all domains, but the level structure of the global database has been retained. The "blackboard" is used as a repository for a domain model, for the representation of partial solutions, and for the representation of pending activities. One important way in which HEARSAY-III has been expanded beyond HEARSAY-II is that it provides for the use of two "blackboards". The "domain blackboard" contains the hypotheses and is intended for reasoning within the task domain, while the "scheduling blackboard" is used exclusively for schduling. Further subdivisons are possible. The "blackboard" supports the construction of labeled graphs consisting of structured nodes called "units" and labeled arcs called "roles". Since the "blackboard" in HEARSAY-III is implemented on top of a relational database system, any relationship to the specific problem being solved can be constructed. The "scheduling blackboard" allows the scheduling process to be broken down, just as the rest of the system is, into a set of independent KSs, each containing its own knowledge about factors that are important to consider in deciding how the system should best expand its resources.

Much of the domain-specific knowledge for an application built in HEARSAY-III is embodied in "knowledge sources (KS)". Each KS can be thought of as a large-grained "production-rule": it reacts to "blackborad" changes produced by other KS executions and in turn produces new changes. To define a KS, three program-units must be provided: the "trigger-pattern", a predicate composed of AND and OR operators, the "immediate code" of the KS which may associate information with the activation record, the "body" of the KS which is run in the triggering context and with the pattern variables instantiated. Each KS execution is indivisible: it runs to completion and is not interrupted for the execution of any other KS activation. This insulates the KS execution and simplifies the coding of the "body"; there need be no concern that during a KS execution anything on the "blackboard" will be modified except as effected by the KS itself.

6.3 HEARSAY Family

In contrast with MYCIN, HEARSAY-II addresses an intrinsically hard problem with characteristics that require particular architectural prescriptions. These characteristics include unreliable data or knowledge, a large search space of possible solutions, inadequate methods for evaluating partial solutions accurately, lack of a fixed sequence of actions that address separate subproblems of the task, the need to guess likely sub-solutions to further the analysis, absence of a strong problem-solving model that could determine effectively which line of reasoning to pursue, the need for inte-

grating diverse bodies of knowledge in the same problem-solving system, and the need for specialized knowledge representations to improve the efficiency of the knowledge application. Because the HEARSAY-II speech-understanding system addresses all of these problems to some extent, its organization as a collection of cooperating, independent specialists, provides a framework for a wide variety of related problem-solving tasks. In fact this kind of design has been adopted for a variety of applications including: signal interpretation [54], cristallography [55], experiment planning [56], task planning and scheduling [57], psychological modelling [58], automatic programming [59], text comprehension [60], and image understanding [61].

SU/X [54]: This is a system that was tested in an application whose details are classified. Its task is the formation and conceptual updating, over long periods of time, of hypotheses about the identity, location and velocity of objects in a physical space. The desired output is a display of the "current best hypotheses" with full explanations. There are two types of input data: the primary signal (to be understood) and auxiliary symbolic data (to supply context for the understanding). The primary signals are spectra, represented as descriptions of the spectral lines. The various spectra cover the physical space with some spatial overlap. The rules given by the expert about objects, their behaviour, and the interpretation of signal data from them are all represented in the IF...THEN form. The situation-hypothesis is represented as a node-link graph, tree like in that it has distinct "levels", each representing a degree of abstraction. A node represents a hypothesis, a link to that node represents support for that hypothesis as in HEARSAY-II, 'support from above' or 'support from below'. Lower levels are concerned with the specifics of the signal data, higher levels represent symbolic objects.

CRYSALIS [55]: This system hypothesizes the structure of a protein from a map of electron density that is derived from X-ray crystallographic data. The map is 3-dimensional, and the contour information is crude and highly ambiguous. The interpretation is guided and supported by auxiliary information, of which the amino acid sequence of the protein's backbone is the most important. Density map interpetation is a protein chemist's art. The automation of this task would require a computational system that could generate its own structural hypotheses, as well as display and verify them. This capability requires: a) a representation of the electron density function suitable for machine interpretations, b) a substantial chemical and stereochemical knowledge base, c) a wide assortment of model building algorithms and heuristics, d) a collection of rules and associated procedures for using this knowledge to make inferences from the experimental data, and 4) a problem-solving strategy for applying these knowledge sources effectively, so that the appropriate procedures are executed at the times that they are most productive. A

problem-solving paradigm that meets the above specifications, to a large degree, is the "blackboard" architecture of HEARSAY-II, specifically with respect to the issues of knowledge integration and focus of attention. A number of different knowledge sources (facts, algorithms, heuristics) cooperate when working on various descriptions of the hypothesis. To use the knowledge sources effectively, a global database - the "blackboard" - is constructed that contains the currently active hypothesis elements at all levels of description.

PLANNING: Although the HEARSAY-II framework was developed around an understanding framework, many of its principal features were extended to develop a model of planning [57]. While understanding tasks require "interpretive" or "analytic" processes, planning belongs to a complementary set of "generative" or "synthetic" activities. The principal features of the HEARSAY-II system which make it attractive as a problem-solving model for speech understanding also suggest it as a model of planning. The planning application shares all the principal features of the HEARSAY-II system, but it also differs in several important ways. In particular, the designers found it convenient to distinguish five separate blackboard "planes", reflecting five qualitatively different sorts of decisions. The "Plan plane" corresponds most closely to HEARSAY-II's single blackboard, holding the desicions that combine to form a solution to the planning problem, i.e., what low-level operations can be aggregated to achieve the high-level outcomes of the plan. These kinds of decisions in generative tasks can be thought of as the dual of the successively higher level, more aggregrated hypotheses constituting the blackboard for interpretation tasks. In the speech task, corresponding hypotheses express how low-level segments and phones can be aggregated to form the high-level phrases and sentences intended by the speaker. The other four planes of the planning blackboard hold intermediate decisions that enter into the planning process in various ways. For example, based on the HEARSAY-II experience with selective attention strategies, resource allocation strategies were formalized and associated explicitly with an "Executive plane".

OTHER APPLICATIONS: Several interesting applications that transfer the approach to other interpetation problems have been made. The HEARSAY-II framework was proposed as a model for human reading behaviour [58]. In this application only one blackboard plane is used, the levels closely approximate those used in the speech-understanding system task, and many additional KSs are introduced to represent how varying amounts of linguistic and semantic knowledge affect the reading skills. HEARSAY-II was further used in a learning system [62] that develops multilevel models of observed game behaviours, and systems were developed that mirror the HEARSAY-II speech understanding components in the image-understanding task [61,63]. Its incremental problem-solving multilevel structure was proposed as a basis for neuroscience

models, and it is considered by distinguished researchers as a source for theoretical psychology fulfilling their intuitions about the form of a general cognitive processing structure. Finally, the HEARSAY-II structure was adapted to the task of interpeting human-machine communication dialogue [64]. Several researcher have focused their efforts on generalizing, refining, or systematizing aspects of the HEARSAY-II architecture for wider applications. A system was developed that assists a programmer in developing a new special-purpose variant [60], and a more formalized, domain independent version has been applied to an automatic-programming task [59], and the planning of genetic experiments [56].

7. PROSPECTOR

The PROSPECTOR mineral-exploration consultation system [65] (see Fig. 7.1), designed for problems in regional resource evaluation, ore deposite identification, and drilling site selection, attempts to represent, like other systems such as MYCIN [27a] or INTERNIST [66], a significant portion of the knowledge and the reasoning processes of experts working in a specialized domain The main function of PROSPECTOR is to match data from a particular situation against models that describe a moderately large number of disjoint classes of situations. To develop a model requires both scientific understanding of the physical and chemical processes of ore deposition and geological judgment based on informed experience. In PROSPECTOR's domain, the models are formal descriptions of the most important types of "ore deposits" which were developed in collaboration with several experienced geologists, and the data are primarily surface geological observations. The available data are assumed to be uncertain and incomplete, so that the conclusion is expressed as a probability or a degree of match. The program also alerts the user to different possible interpretations of the data and identifies additional observations that would be most valuable for reaching a more definite conclusion. PROSPECTOR recently made a prediction about the location of molybdenum ore at an exploration site in the state of Washington. The prediction was substantially confirmed by drilling with a finding worth 100 million dollars.

7.1 PROSPECTOR System

The knowledge base 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 knowledge-base encodes as much as possible of the background knowledge that is useful for several applications and situations of the domain. It is organized around models of different "ore deposits" including "Mississippi Valey lead and zinc", "Komatiitic nichel sulfide", "Yenington prophyry", and others (see Fig. 7.2). The special purpose knowledge-base encodes statements that are relevant to some specific subset of the domain and contain primarily the inference networks (rules and other inference structures) in which these statements participate. All elementary domain-specific notions are in PROSPECTOR integrated in "taxonomical tree structures"; the nodes, representing the simple concepts of the domain, are connected by arcs which indicate the element (e) and subset(s) relationship 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. In Fig. 7.3 we show examples of taxonomies in PROSPECTOR's knowledge base including rock

types, minerals, physical forms, and geological ages. Each node X in the hierarchical structure is said to be a restriction of its parent nodes or of any node occuring on a chain of outgoing "e" and "s" arcs from X. 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, (expressed by "e" and "s" arcs), are supplemented by the more restrictive concepts of "disjoint subsets (ds)" and "distinct elements (de)". 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, de-arcs indicate that each of two or more nodes denotes a different element of a set.

This 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 network described below). "Partitioned semantic networks" [67] are used to encode statements in the knowledge base. Each statement is represented by a "structured object" (called "unit") where the semantic representation of the statement is strored in terms of primitive relations and entries in the various taxonomies of the domain. Let us illustrate this with the example: "a rhyolite plug is present", which is presented in Fig. 7.4 as a small network that makes the following three assertions:

(a1) there exists a physical entity E1,
(a2) the composition of E1 is "rhyolite"
(a3) the form of E1 is "plug"

In general, an assertion corresponds to a node inside the space (delimited by the rectangle in Fig. 7.4) that constitutes the semantic representation of the statement. Because "rhyolite" and "plug" can be referred to from other statements of the knowledge base, they are not included in that space, but appear instead as entries in the taxonomy of "rocks" and the taxonomy of "forms", respectively. These "external references" are most frequently 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 knowledge base. In addition to physical entities, a variety of other concepts such as places (locations) and geological processes are described in the knowledge base. The attributes associated with a concept appear in the semantic representation as relations of two or more arguments. COMP-OF(,) and FORM-OF(,) are two common attributes of physical entities; others are: AGE-OF, GRAIN-SIZE-OF, LOC-OF, etc. . 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 attributes are external references but can also be other concepts included in the statement being described.

The data structures for representing the geological know-

ledge embodied in PROSPECTOR is called "inference network" and guides the plausible reasoning performed by the system. The nodes in this network correspond to various assertions such as: "there is pervasively biotized hornblende (≡E)", or "there is alteration favorable for the potassic zone of a porphyry copper deposite (≡H)". Most of the arcs in the inference network define "inference rules" that specify how the probability of one assertion affects the probability of another assertion. For example, from ⟨E⟩ follows ⟨H⟩, or the absence of ⟨E⟩ is very discouraging for ⟨H⟩. In a particular run, any assertion may be known to be true, known to be false, or suspected to be true with some probability. An inference network is equivalent to a collection of "inference rules". In general, any inference rule has the form

IF ⟨E⟩ THEN* ⟨H⟩
*(to degree LS, LH)

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, LH 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 parameters thus specify the sufficiency (LS) and the neccessity (LH) measures, respectively, and must be supplied by the domain expert 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 different hypotheses. Fig. 7.5 shows a portion of the inference network encoding of a PROSPECTOR model for "porphyry copper deposits". PROSPECTOR's knowledge base contains 15 models with over 1000 rules and 1500 "units". The taxonomies shared by all models contain over 1000 entries.

Although the assertions (meaning Evidences and Hypotheses) are statements that should either be true or false in a given situation, there is usually uncertainty as to whether they are true or false. Initially, the state of each assertion is simply unknown. As evidence is gathered, some assertions may be definitely established, whereas others may become only more or less likely. In general, we associate a probability-value with every assertion. The "connections" in the inferen-

ce network determine how a change in the probability of one assertion will affect those of other assertions. The principle or top-level assertion in an inference network for a model is the assertion that the available evidence matches that particular model. To establish this assertion, it is usually necessary to establish several major factors. For example, to establish the top-level assertion in one of the models (called Model-II), we must establish the following hypotheses:

1. The petrotectonic setting is favorable for Model-II,
2. The regional environment is favorable for Model-II
3. There is an intrusive system that is favorable for Model-II.

If any of these assertions were a field-observable evidence, it could be established merely by asking the user of the program whether they were true. However, since all of these factors are hypotheses, each must be further related to other factors. For example, the favorability of the "petrotectonic setting" can be established through the following three factors, each of which happens to be determinable (at least in principle) from observational evidence:

1. The prospect lies in a continental margin mobile belt;
2. The age of the belt is post-Paleozoic;
3. The prospect is subject to tectonic and magmatic activity related to subduction.

In general, the ore deposit models in PROSPECTOR have this type of hierarchical structure. The top-level assertion is determined by several major second-level assertions, each of which may be determined by third-level assertions, with this refinement continuing until assertions are reached that can be established directly from field evidence. This is shown in Fig. 7.5 where the regional environment of Model-II is decribed. In addition to this "top-to-bottom" development in terms of successive levels of assertions, the models also often exhibit a "left-to-right" organisation in terms of spatial scale, from the petrotectonic setting on the left to the local details of mineralization and texture on the right. Exactly how these considerations interact is determined by the relations that exist among the assertions.

Three basically different kinds of relations are used in PROSPECTOR to specify how a change in the probability of one assertion affects the probability of other assertions. We distinguish these as "Logical Relations", "Plausible Relations" and "Contextual Relations". With "Logical Relations", the truth (or falsity) of a hypothesis is completely determined by the truth (or falsity) of the assertions that define it. Such relations are composed out of the primitive operations of conjunction (AND), disjunction (OR), and negation (NOT). In general we do not know whether the assertions are true, but can, of course, estimate a probability or degree of belief that they are true. With "Logical Relations", to compute the probability of a hypothesis from the probability of its component assertions, the fuzzy-set formulas of Zadeh [68]

are used. Using these formulas, the probability of a hypothesis that is defined as the logical conjunction (AND) of several pieces of evidence equals the minimum of the probability values corresponding to the evidence. Similarly, a hypothesis, defined as the logical disjunction (OR) of its evidence "units", is assigned a probability value equal to the maximum of those values assigned to the evidence "units". With "Plausible Relations", each assertion contributes "votes" for or against the truth hypothesis. This would be expressed by relating the assertions to the hypothesis through a set of plausible inference rules. Each rule has an associated rule strength that measures the degree to which a change in the probability of the evidence-assertion changes the probability of the hypothesis. This change can be positive or negative, since an assertion can be favorable or unfavorable for a hypothesis. As with all parts of the model, these rule strengths are obtained by interviewing an authority on the corresponding class of ore deposits. This information is translated into numerical terms (as shown in Fig. 7.5), the changes in probability being computed in accordance with the rules of Bayesian probability theory. 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 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 domain expert at the time the model is constructed, but can also, in some cases, be computed from the prior-probabilities assigned to the related "units". The "Conceptual Relations" take into account that assertions cannot be considered in an arbitrary order, but must be considered in a particular sequence. For example, the existence of a "continental margine belt" would be specified as a context for asking about the age of the "belt". Thus, before inquiring about the age, the system would employ all its resources to establish the existence of the belt, and would not ask about its age unless the probability of the belt were greater than its initial value. Contextual relations are also used when one assertion is geologically significant only, if another assertion has already been established. In such instances it would not be useless to ask the former question without first establishing the latter, but it is the case that the former evidence is geologically irrelevant without the latter to establish a match to the model. Two such instances are depicted by the dashed arrows in Fig.7.5.

PROSPECTOR is a mixed-initiative system that begins by allowing the user to volunteer information about the geological prospect. This volunteered information is currently limited to simple statements in constrained English about the names, ages, and forms of the rocks and the types of minerals

present. These statements are parsed by LIFFER [69], a natural interface facility, and represented as "partitioned semantic networks". A network matching program compares each of these volunteered "units" against the "units" in the models, noting any subset, superset, or equality relations that occur. If a volunteered "unit" is exactly equal to a "unit" in a model, the probability of the model-"unit" is updated and that change is propagated by forward-chaining through the inference network. If a volunteered "unit" is a subset of a "unit" in a model and if it has a higher probability than the model-"unit", once again the probability of the model-"unit" is updated and that change is propagated through the inference network. Unfortunately, if the volunteered "unit" matches a superset of a model-"unit" no probability change can be made unless the user expresses doubts about the situation. For example, if the user mentions "biotite", the probability of the "unit" that asserts that there is "pervasively biotized hornblende" is unchanged, unless the user has said that he doubts that there is any "biotite". However, it is obvious that the system may want to follow up this observation, and the existence of the connection to the model is recorded. When the user has finished the initial volunteering, PROSPECTOR scores the various models on the basis of the number and types of connections that have occured and selects the best matching model for further investigation. From here on the basic control strategy is MYCIN-like "backward-chaining". At any given time there is a "current-goal unit" whose existence is to be determined. The initial goal "unit" is the one that corresponds to the best matching model. The various "units" in the models represent either evidence that can be sought form the user (are "askable") or internal hypotheses that are to be deduced from evidence (are "unaskable"). Naturally, the initial goal "unit" is always unaskable. If the current goal "unit" has any unestablished "context-units", they are pushed on the goal stack and one of them becomes the current goal. If the current goal is askable and has not been asked before, the user is asked about it, the effects of the answer are propagated througth the inference network, and the process is repeated. If it is unaskable, it must be either the consequence of one or more inference rules or a logical combination of one or more other "units". In the former case, the rules are scored to determine their potential effectiveness in influencing the Hypothesis ‹H›, and the Evidence ‹E› (or antecedent) of the best-scoring rule becomes the next goal. In the latter case, a predetermined supporting "unit" becomes the next goal. In either case, the same procedure is repeated until a) the top-level goal becomes so unlikely that another top-level goal is selected, b) all of the askable "units" have been asked, or c) the user interrupts with new volunteered information.

7.2 KAS (Knowledge Acquisition System)

The KAS-system [70] is the PROSPECTOR consultation-program

without its domain-specific knowledge. Related to PROSPECTOR in basically the same way, as EMYCIN is to MYCIN, KAS allows the PROSPECTOR inference and control mechanisms to be used on new problems when the domain-specific knowledge can be represented in the KAS rule language. The KAS/PROSPECTOR inference engine is distinguished from EMYCIN mainly by performing "forward- and backward-chaining". This allows significant changes in the choice of the "high-level goals" to occur in response to information acquired. KAS tries to avoid low-payoff questions by using a heuristic evaluation function to choose the most promising rules. KAS'use of variables is severely restricted. EMYCIN allows repeated use of the rules by creating different instantiations of the objects in its ‹attribute, object, value› triplet, whereas KAS allows only one for which the situation description by the triple is thought to be most certain. This characteristic allows KAS to interconnect its rules into a static network before run-time, thereby eliminationg the need for searching through the rules to propagate inferences.

The approach in KAS is 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. The first layer was selected to contain general knowledge about networks. The subsequent layers contain further specification of the various kinds of network, and knowledge about every component that constitutes the resulting expert system, including the inference procedures, the consultation systems, and finally knowledge about the domain of application. The knowledge acquisition tools, such as the resident network editor and the book keeping system 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 appropiate layer. The advantage of this layered structure is that knowledge acquisiton tools can be designed efficiently and with a high level of generality to assist in different phases of the knowledge acquistion process, where in each phase only some aspect of the knowledge base is of interest or available to the knowledge engineer. 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 each component of the expert system.

In KAS there is no formal top-level goal, but certain "consequents" are distinguished as "top-level hypotheses". A consultation narrows down continually this list by establishing the truth or falsity of the most promising ones. If no clear conclusions can be reached, the system identifies the missing information for resolving the situation. Thus KAS either tries to identify the best top-level hypothesis which is worth to be pursued (goal-selection mode) or it questions the

user in order to establish that hypothesis (question-asking mode).

The "goal-selection mode" is guided by the user information. Initially any relevant information is volunteered in the form of simple English statements. It is parsed and integrated in the semantic networks, whereby each assertion is matched against the descriptive statements in the knowledge base. If partial or exact matches are detected, the system updates a heuristic score for that top-level hypothesis which is supported by that statement. This scoring function takes into account the certainty of the evidence, the nature of the match, and , by tracing through the rules, whether the evidence is favorable or unfavorable for the hypothesis. The best-scoring hypothesis is usually persued further by the system. It however also supports mixed-initiative control by allowing the user to overrule the system in selecting the current goal hypothesis ‹H›.

If the goal-hypothesis ‹H› has been determined, the program switches into the "question-asking mode". The statements lying one level below ‹H› are inspected in order to find the one that has the strongest influence on the probability of the statement ‹H›. Let ‹S› denote this statement. If ‹S› is marked as "askable" and if the user has not been asked about ‹S› previously, the system asks whether ‹S› is true, forward-chains to propagate the consequences of the answer, and returns a new goal, and the same procedure is reapplied, resulting in "best-first backward-chaining". How is the statement with the greatest effect on ‹H› determined? First, the required "contexts" of ‹H› are immediately established by a new goal. The next step then depends upon whether ‹H› is a "logical statement" or a "descriptive statement". For "logical statements", a special procedure chooses the least likely unexhausted argument for disjunctions. This procedure fails if all the arguments are exhausted or if the certainty of the logical expression is appropriately bounded. For "descriptive statements", another special procedure uses heuristic criteria to score the rules having ‹H› as a "consequent". The scoring function takes into account the current certainty of both the "antecedent" and the "consequent", as well as the strengths of the rules.

KAS provides a well-engineered environment for developing and debugging rules, semantic networks, and taxonomic structures. Its knowledge-base editor has three particularly valuable features: 1) it operates directly on network structures, 2) it uses knowledge about representational formalism to assist the knowledge-base designer, and 3) it facilitates the development by allowing the designer to get immediate feedback on the consequences of changes to the knowledge base. For details we refer to Ref. [70].

8. APPLICATION DOMAINS

In this section we give some insight into the diversity of the application domains for expert systems. In the first part we focus on the different tasks [2], and in the second part we give an overview about some of the present day expert system activities [71].

There are several generic tasks that experts can perform, and examining these tasks will provide us with a guide for the requirements and preferred architectural structure for an expert system of a particular task. We distinguish in particular: interpretation, diagnosis, monitoring, prediction, planning, and design, as well as debugging, repair, instruction and general control.

INTERPRETATION: the analysis of data with a specific interpretation goal in mind. The interpretation of the mass-spectrometer data, as is done for instance in the DENDRAL-system [32], is a typical example. The data are measurements of the masses of molecular fragments, and the interpretation means the determination of one or several chemical structures. The main task is to find the correct and consistent interpretations of the data, whereby one expects a rigorous and complete analysis of all possible interpretations with potential candidates being ruled out only when there is enough evidence to do so. The data are often noisy and erroful which means that there might be missing or erronous or even extraneous values. The interpretation system must therefore cope: 1) with partial information, 2) with data that seem contradictory, requiring a hypothesis on which of them are correct, 3) with unreliable interpretations of the system, and 4) with the requirement to get information about the sometimes rather long reasoning steps.

DIAGNOSIS: the process of fault-finding in a system mostly based on uncertain and noisy data. The MYCIN-system for infectuous blood-diseases [27] is a typical example. It is necessary that the system's organization is understood, including the relations and interactions between its sub-systems. Some of the key problems arise from the fact that: 1) the faults are sometimes masked by the symptoms of other faults, 2) the faults can be intermittent, 3) the diagnostic equipment itself fails, 4) some data are inaccessible, or 5) the function of the system itself is not fully understood.

MONITORING: this means the continous interpretation of signals to set off the alarm when an intervention is required. There exist a few computer-aided monitoring systems, still in the research stage, for nuclear plants [72], for air traffic control as well as for disease [37-39], regulatory [40-43] and fiscal management goals. As a representative example we mention the VM-system [40] which is a physiological monitoring system designed to: i) detect possible measurement errors, ii) recognize untoward events in the patient/machine system and suggest corrective action, iii) summarize the pa-

tient's physiological status, iv) suggest adjustments to the therapy based on the patients status over time and long-time therapeutic goals, and v) maintain a set of case-specific expectations and goals for future evaluation by the program. A monitoring system recognizes the alarm conditions in real-time. The key difficulty arises from the recognition of an alarm condition which is often context-dependent. To account for this key problem, monitoring systems have to vary signal expectations with time and situation.

PREDICTION: the future course of a system is predicted, based on a model of the past and present. Examples of this category are: wheather forecasts, demographic and traffic predictions, military forecast, voting forecasts and so on. A prediction system typically employs a parametric dynamical model with its parameter values fitted to a given well-understood situation. The consequences, inferable from a model, then form the basis of the predictions. By ignoring probability estimates, the prediction systems can generate large numbers of possible scenarios. Prediction requires reasoning about time, meaning that the predictors must be able to refer to things that change over time and to events that are ordered in time. There are several key problems that can arise: 1) the prediction requires the synthesis of incomplete information, 2) it must acount for a spectrum of possible sitations in the future, whereby variations in the input data is likely to occur, 3) the data must be diverse since the indicators for the future may appear at quite different places, 4) the predictive theory may need to be contingent since the likelyhood of distant futures may depend on the nearer but unpredictable events.

PLANNING: it consists of the preparation of program-actions to be carried out to achieve a goal. Typical systems involve automatic programming as well as robot, project, route, communication, experiment, and military planning problems. Planning systems emply models of the agent behaviour to infer the effects of the planned agent activities. As a typical example we mention the MOLGEN-system [56], a knowledge-based program that helps molecular geneticists in planning experiments. A planing system constructs a plan that achieves goals without consuming excessive resources or violating constraints; it establishes priorities if some goals conflict and it must be flexible and opportunistic since the planning requirements can change with time or decision data which are possibly incomplete. Planning always involves a certain amount of prediction. Some of the arising problems are: 1) the consequences of a planned action can often not be forseen, and consequently tentative action is most appropriate, 2) if there are many details, focus on the most important problems is required, 3) in large, complex systems there can be interactions between the plans for different sub-goals which must be localized and taken into account, 4) since the context in a planning goal is not fully known there is quite some uncertainty involved, and 5) if there are multiple ac-

tors, there is need for coordination.

DESIGN: this is the configuration of objects that satisfy the constraints of the design problem. Circuit-layout, building-design, budgeting, etc. are typical examples; the EURISKO-system [73], which ranges in this category, has been successfully applied to the VLSI-design. Many design-systems are built to minimize an objective function that measures costs and other undesirable properties of potential designs. This view of the design problem can also subsume the goal-setting behaviour, with the objective function incorporating measurements of goal attainment. The design has often similar requirements as reasoning. Several types of difficulties can arise: 1) the design possiblities must be explored tentatively, 2) the design constraints can come from many sources, 3) a large design problem is preferably factored into sub-problems whereby possible interactions may not be ignored, 4) a large design system should keep a justifying record about the design decisions as a mean to memorize the design decisions, 5) whilst redesigning parts of a program the influence on the overall program must be taken into account preventing locally relevant modifications only, 6) reasoning about spatial relations demands considerable computing resources since the effective techniques about this type of problems are still research topics.

There are a few other categories which are closely related to those discussed above such as:

DEBUGGING: such computer-systems prescribe remedies for the mal-functioning of a task-system; they rely on planning, design and prediction capabilities to create specifications of recommendations for correcting a diagnosed problem.

REPAIR: the computer-systems of that type develop and execute plans to administer a remedy for diagnosed problems; they incorporate debugging, planning and execution capabilities.

INSTRUCTION: the computer-systems of that type are intended as teaching assistance since they diagnose and debug the students behaviour. They begin by construction a hypothetical description of the students knowledge, subsequently they spot the incompleteness in the students knowledge and identify possible remedies, and finally they plan a tutorial action.

CONTROL: the expert control system adaptively govern the overall behaviour of a complex system. It repeatedly interprets the current situation, predicts the future, diagnoses the causes of the anticipated problems, formulates a remedial plan, and monitors its execution to ensure success.

In the above classification of the expert systems according to their goals and purposes, one comes across several design issues: large solutions space, tentative reasoning, time-varying and noisy data. The large solution spaces usually appear in interpretation, planning and design systems, whereas the tentative reasoning is a typical characteristic in diagnostic, design and planning tasks. The reasoning in real-time, called real-time systems control [72], is quite

often needed in monitoring systems with interpretation, diagnostic and monitoring tasks.

We sofar have limited ourselves to a rather qualitative and quite general classification of the expert systems by considering the, in principle possible, classes of tasks. We now go to the practical side by considering the expert-systems that exist as well as some of the projects currently being carried out at the main research centers. For this purpose we have assembled in Fig. 8.1 [74] an extended list of expert-systems and tools indicating their principle goals; this list is however by no means complete and it keeps rapidly growing. It would go beyond the purpose of this paper to give an introduction or overview about all these systems. We therefore chose to discuss a few important topics giving the reader a flair of the main lines of research being currently pursued and of the related fields of application.

In Fig. 8.2 [71] we show how some of the leading present-day (planned) applications are distributed according to their particular purpose and the industrial need. On the top-horizontal axis the industries and related professions are shown, such as: Engineering, Electronics, Telecommunications, etc., and on the left-vertical axis there are the possible applications. In the following we dwell on a few of these topics:

INTELLIGENT CAD: In this field of design we distinguish several important directions of applications such as for instance configuration systems and VLSI-design.

One of the first areas where expert systems have proved commercially valuable is the computerisation of the design process, with applications ranging from the planning of large buildings to the layout of integrated circuit. By comparison with "conventional" computer-aided design (CAD) techniques, symbolic computing with the expert system tools has several advantages:

1) The symbolic languages allow a more direct representation of the design concepts. These include the objects being manipulated, the rules governing their arrangement, the dependencies which link them, and the contraints which limit them.
2) The expert-systems methodology makes it much easier to cope with uncertainty. By definition, any design process works from a starting point where the ultimate result cannot yet be defined. It is usually necessary to strike a balance between the objectives the design is intended to achieve and the resources available. An exploratory approach is needed to find an optimal result.
3) A knowledge-base provides a suitable means of representing design expertise, much of which is heuristic and informal. The development of a common knowledge base can provide an important resource in itself for the user community.

This is an area where the computer industry itself is in the lead. DEC's XCON/R1-system for configuring VAX computer-system orders is an often cited example. XCON/R1 does all the work of deciding whether the order is valid and complete as it stands and how all the components can be fitted together as a working system. The system uses a forward-chaining strategy. A complete configuration design is divided into six tasks, from the initial check whether there are any major errors in the order to finally working out the cabling requirement. Tasks are further subdivided into a total of nearly 300 sub-tasks, typically 5 to 15 rules each. The program can search this relatively small description space without facing a combinatorial explosion. XCON/R1 starts each task from a partially complete configuration and then extends it step-by-step until it is complete, according to rules provided in the knowledge base. Once the task is complete it proceeds to the next, following the same sequence each time with no provision for backtracking. DEC's XCON-system is currently being complemented by several other, similar expert systems called: XSEL, XSITE, ISA, IMACS and ILOG. Digital's "Expert Sales Assistant System (XSEL)" is intended to provide the field sales force with help in configuring customer systems, planning the site requirements and floor layout, and estimating delivery dates. The "Expert Site Planning System (XSITE)" is intended to complement XCON and XSEL at the other end of the sales cycle by helping the customer service department to prepare for the installation and maintenanc of computer systems. The "Intelligent Scheduling Assistant (ISA)" schedules customer orders against current and planned materials allocations, customer credit lines, and planned delivery dates. The "Intelligent Management Assistant for Computer Systems (IMACS)" provides management assistance in planning the assembly and test of computer systems, paperwork, capacity planning, floor loading, throughput, testing plans and inven-tory control. The task of the "Intelligent Logistics Assistant (ILOG)" is to generate plans for the distribution of computer systems to customers. Together these five systems represent only a part of DEC's current work in applying expert systems technology to its own business. The company is currently working on a total of nine cooperating systems which could ultimately provide integrated assistance in managing the company as a whole. IBM and NCR are two other computer manufacturers known to be developping configuration systems, and ICL already has a sizing system called DRAGON in a well developed state.

VLSI-DESIGN: The integrated circuit industry is also making early use of "intelligent CAD". Helped by the close proximity of several Stanford academic institutions and the related AI centers to the semiconductor component industry in Silicon Valley, firms such as Xerox, Fairchild Camera, Hewlett Packard and Daisy Systems are known to be applying expert systems techniques to raise the quality and productivity of the VLSI design. No doubt most of the other leading semiconductor firms are doing the same. The work on applying expert system technology to VLSI design can be seen as a two step process.

A number of companies have taken the first step and are using AI programming environments as the most productive means of developing the highly complex software required for a VLSI design workstation. We mention the NEWDRAW logic design system (by LISP Machine Inc.) as an example. NEWDRAW itself makes use of AI concepts to improve designer productivity. For example, it can allow considerably more flexible use of the conventional design hierarchy, descending from highlevel block diagrams where each individual block may itself contain a block diagram, down to the level where individual components are represented. In a conventional system, the designer can define a component and use it repeatedly throughout the hierarchy without needing to redefine it, but if it needs small changes in different contexts it is necessary to define each version as an entirely new object. An "intelligent CAD" system, such as NEWDRAW, can allow the user to define a generic type of object, and then to modify it in the context where it is used. This can allow considerable savings in design time in tailoring a circuit to fit a requirement precisely, or changing the characteristics of a generic type throughout a design. Stanford University is one of the leaders in taking the natural next step by developing expert systems for logic and VLSI-design, in the EURISKO and PALLADIO projects [74]. EURISKO is a very generalized system which is able to make discoveries and formulate hypotheses in an assign domain. It has been used to explore the possiblities of 3-dimensional microelectronic circuits - todays semiconductor chips have an essentially planar, 2-dimensional structure - and it was able to propose some novel and potentially useful devices. Among other things, the experiment showed the importance of selective search strategies guided by heuristic rules. Running EURISKO with an exhaustive search strategy, it was able to synthesize a possible 3-dimensional device every 0.9 seconds, but the proportion of good devices was estimated at below one in a billion. Using a search strategy guided by about 100 heuristic rules it took 30 seconds to synthesize a device design, but the proportion of some interest rose to one in ten. EURISKO is still a long way short of designing complete circuits, but clearly it could help a designer to explore a wider range of new ideas and possiblities than before. PALLADIO is more specialized, providing an exploratory environment for integrated circuit design. It allows the designer to define models, called "perspectives", at different levels of circuit design, ranging from the specification of the masks used to fabricate the circuit on silicon to the high-level design of novel logic architectures. Expert system aids assist in translating an abstract design into more concrete specifications; for example by specifying the interconnection requirements for a circuit described in terms of switches and gates.

REAL-TIME MONITORING: A great deal of AI work has been concerned with "signal understanding" systems, typically aiming to operate in real time. The biggest area of all is na-

tural and spoken language understanding. Language developments such as the HEARSAY-II system (at Carnegie Mellon University) have been an important source of the programming tools and architectural ideas which are being used as the basis of real-time expert systems in other areas.

A "blackboard" architecture is the common basis of most if not all real-time expert systems. The blackboard provides an effective interface between a continuous flow of data from many sources - local databases, online information retrieval, physical sensors and monitors - and the knowledge base which must monitor events. Typically, incoming data is posted to a hierarchy of levels on the blackboard and monitored by a number of expert system modules called "knowledge sources". The function of the KSs is to draw conclusions from the data inputs at their level of the blackboard and forward the conclusions up to higher levels so that the system as a whole can achieve a wider undertanding of events in the domain it is observing, and report on it, or suggest actions to its users.

Three major aplication areas where real-time expert-systems using blackboard architectures are seen as having special value in the short term are: i) military command and control, ii) proecess control of industrial plants, iii) foreign exchange and commodity trading. Beyond these applications, there is a very wide long-term potential for the blackboard architecture approach. Civilian air traffic control, driving vehicles, perhaps the control of robots with something of the agility of humans, are possiblilities which are seriously considered.

Among the "US military expert-systems" currently being developed there are three major application areas: autonomous vehicle, operational associates, battle management.

The "autonomous vehicle application" envisages systems able to control land, submarine, air and space vehicles quite freely without human intervention. For example it might be used to control tracked land vehicles travelling through hazardous battle zones for reconnaissance, the re-supply of forward positions, or ammunition handling.

The "operational associate" would have the different role of assisting human combatants to control their machines - the examples suggested are pilots and tank crew. It would be trained, by a pilot for example, to act as a personal assistant in making split-second flying decisions with performance requirements about 100 times faster than current technology.

The "battle management application" would assist the commander in different types of battle to assimilate the mass of incoming data, identify threats, and decide how to counter them. Versions are foreseen for land battles at battalion level, naval fleet battle, ballistic missile defence and the control of adaptive hardware for electronic counter-measures to radio jamming effects. First systems of this type exist

such as for instance AIRPLAN which assists air operations officers with the task of landing an aircraft on a naval carrier, or the "Signal Understanding System(SUS)" which has been conceived to monitor the wide variety of different types of information available to the commander of a naval vessel at his control centre. SUS is able to combine these varied inputs to provide a real-time display of the ships, submarines and aircraft in a 200-mile zone around the vessel, tagging them as hostile or friendly, and providing immediate warning of an imminent threat. SUS is a classic blackboard architecture system, with 35 knowledge sources monitoring and analysing signals and passing messages up to the level where threats can be identified.

The expert-systems for "Industrial Process Control" can be adapted from those for military objective. The task for SUS is required, is directly parallel in the monitoring and control of an industrial plant. Instead of continuous radar input, the plant operator has continuous instrument readings, instead of military intelligence he has personal observations and reports from other staff, instead of military plans there are operating schedules, and so on. Clearly, expert-systems could be very significant for the control of nuclear power stations and other plant requiring very high safety standards. Although it has yet to be demonstrated in practice, expert-systems could lead to a significant improvement in the safety and responsiveness of process control.

Another field is "Foreign Exchange and Commodity Trading". Speed in the sense of immediate response to immediate information is vitally important to banks and brokers in the foreign exchange and commodity markets. Again, the blackboard architecture is an appropriate way to combine a numer of different sources of data - the incoming news service, the traders client database, knowledge bases about the character of each market and about the business of trading. One problem about integrating these streams of data at the computer level is that the most advanced news service (by Reuters) is provided only in analogue video form; it however is unlikely that this will remain a barrier for long.

MANAGEMENT SUPPORT: Some of the biggest and most ambitious expert systems yet undertaken for commercial applications are aimed at providing management assistance of some kind. Carnegie Mellon University (jointly with some computer companies) has specialized in this area, aiming to develop systems which tackle directly the major tasks of managing a large organisation, and which can be linked together to provide an "Integrated Management Systems (IMS)". This is envisaged to be able to: i) sense the state of the plant under management by automatically acquiring datá, ii) representing the status of all the objects, machines, people and concepts involved, iii) model the organisation at many levels of abstraction, iv) analyze and manipulate the model to answer management inquiries, and v) anlyze how the performance of the organization could be improved by changes. Some of the earlier men-

tioned expert-systems by DEC are being developed as part of the IMS project.

Another major project has been the ISIS-system. The goal has been to produce an operational prototype system for the management and control of production in an engineering job-shop (such as for instance Westinghouse turbine-component plant), and so to investigate the application of AI techniques to this type of management task. In a jobbing shop, components are produced in batches which are largely unpredictable in size, timing or detailed specification. Their production is subject to many different constraints, and a major reason for the limited success of conventional computer scheduling packages in this area has been their inability to take account of all the constraints which apply. Thus the plan they produce can be no more than suggestions, subject to change on the factory floor when unscheduled constraints intervene.

FINANCE SERVICE: Several new venture companies in the field of expert-systems are targeting on financial services as a major or exclusive application area. The expert-systems applications identified by most financial service companies are: i) credit extension, ii) financial planning for clients, iii) asset and liability management for the company itself, iv) foreign exchange operations, and v) insurance underwriting. Of these is the "portfolio management" functions which fit most naturally into the management support category. As financial institutions on both sides of the Atlantic become more diversified, the advantages of being able to provide a genuine portfolio management service to middle-market clients are becoming increasingly obvious. One way in which expert-systems technology can help to solve the problem is illustrated by a system called FOLIO. This is a small expert system which uses a 50-rule forward-chaining knowledge base as the means of automatically assessing the client's investment goals. This is combined with a simple interview system to gather data from the client, and a linear programming algorithm to optimise the distribution of assets between nine different fund types once the goals have been determined. Fund types are broad categories such as "high-income, low-risk stocks" or "government bonds"; it remains the job of the investment advisor to select particular securities within these categories. The management of a bank' ownd portfolio raises similar problems on a much bigger scale. The bank must continually seek to match its assets and liabilities to maintain its liquidity while earning a profit. A strategy for doing this has to take into account the bank's views on the future trends of interest rates and the funds it has available. Not only are these subject to change, but they are always uncertain, and the bank must ensure that it is covered even if the outcome is considerably less favourable than expected. The task of funds managment in this environment is one of developing alternative strategies and evaluating their future results against a range of possible scenarios.

9. SUMMARY

In this paper we have given an introduction into the "world" of expert-systems and knowledge engineering by discussing a few typical examples: starting from the basics of "knowledge-representation", subsequently considering, in somewhat more detail, a few concrete expert-system programs and their specific characteristics, and finaly widening our horizons with a glimps on the present-day and the planned activities in a few selected domains.

Knowledge engineering offers a spectrum of exciting activities ranging from : the development of new expert-system tools with better knowledge-representation formalisms, search strategies, user facilities and so on; to the actual construction of a concrete system with domain-specific knowledge; and to the search and selection of new useful applications. This latter issue is in particular always in the back of the author's mind, and the aim of this paper is therefore a deeper understanding of these knowledge manipulation systems.

Obviously, this presentation is incomplete: the frame, script and procedural representations and their corresponding computer systems could not even be touched upon; the third part of this paper, concerning the advanced present-day activities, is fragmentic and sketchy. We nevertheless would like to encourage the interested reader to go beyond this last (summary) section in order to form a realistic opinion, free of over-enthusiasm or even euphorism, on the usefulness of the expert-systems.

Acknowledgements

The author thanks Dr. P. Vogel and Prof. C. Joseph for their kind interest and encouragements. The author also thanks the CERN Theoretical Physics Division, where part of this work was done, for its kind hospitality.

REFERENCES

1) A. Barr and E. A. Feigenbaum, The Handbook of Artificial Intelligence I, II, III, Heuristech Press, Stanford, California (1981); and references therein.
2) F. Hayes-Roth, D. A. Waterman and D. B. Lenat, Building Expert Systems, Addison-Wesley, Publishing Company, Inc., Reading, Massachusetts (1983).
3) M. Schindler, Electronic Design 122 (1984) 106
4) For a good overview see for example: J. A. Robinson, Logic: Form and Function, Edinburg University Press (1979).
5) B. Humpert, Theorem Proving with First-Order Predicate Logic I, Proceedings of the 8th Warsaw Symposium on Elementary Particle Physics, Warsaw University Press(1985).
6) J. McCarthy, Programs with Common Sense, Proceedings of the Symposium on the Mechanization of Thought Processes, Natl. Physical Laboratory 1 (1958) pp. 77-84; (Reprinted in M. L. Minsky, Ed.) Semantic Information Processing, Cambridge, Mass. MIT Press (1968) pp. 403-409.
7) J. A. Robinson, Journal of the ACM 12 (1965) pp. 23-41.
8) C. C. Green, Proceedings of the Joint Intl. Conf. on Artificial Intelligence 1 (1969) pp. 219-237.
9) R. E. Fikes, P. Hart and N. J. Nilsson, Artificial Intelligence 3 (1972) pp. 251-288.
10) R. E. Filman and R. W. Weyhrauch, An FOL Primer, Memo 288, AI Laboratory, Stanford University (1976).
11) D. Prawitz, Natural Deduction: Proof-Theoretical Study, Stockholm, Almqvist and Wiksell (1965).
12) P. C. Gilmore, IBM J. Res. Develop. (1960) pp. 28-35, and ref (8).
13) M. Davis and H. Putnam, J. Assoc. Comp. Mach. 7 (1960) pp. 201-215.
14) C. Chang and R. C. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York (1973).
15) L. Wos, R. Overbeek, E. Lusk and J. Boyle, Automated Reasoning: Introduction and Applications, Prentice-Hall, Inc. (1984).
16) R. S. Boyer and J. S. Moore, A Computational Logic, Academic Press, N.Y. (1979)
17) A. Newell and H. A. Simon, Human Problem Solving, Prentice-Hall, N. J. (1972).
18) E. Shortliffe, Computer-based medical consultations: MYCIN, American Elsevier, NY (1976).
19) R. Davis, in: Knowledge-based systems in Artificial Intelligence, McGraw-Hill (1980) New York.
20) S. M. Weiss and C. A. Kulikowski, Proc. of the Intl. Joint Conf. on Artificial Intelligence 6 (1979) pp. 942-947.

21) D. Lenat, in: Knowledge-based systems in Artificial Intelligence, McGraw-Hill (1980) New York.

22) J. Kunz et al., A physiological rule-based system for interpreting pulmonary function test results, Heuristic Programming Project Rep. No. HPP-78-19, Computer Science Dep., Stanford University (1978).

23) M. R. Quilian, Semantic Memory, in: Semantic Information Processing (Ed. M. Minsky) Cambridge, Mass. MIT Press (1968) pp. 216-270.

24) B. Raphael, SIR: A computer program for semantic information retrieval, in: Semantic Information Processing (Ed. M. Minsky) Cambridge, Mass. MIT Press (1968) pp. 35-145.

25) R. E. Fikes and G. Hendrix, Proc of the Joint Intl. Conf. on Artificial Intelligence 5 (1977) pp. 235-246.

26) R. O. Duda et al., Development of the PROSPECTOR consultation system for mineral exploration, SRI Projects 5821 and 6415, SRI International Inc., Menlo Park, California (1978).

27a) see Ref. (18).

27b) B. G. Buchanan and E. H. Shortliffe, Rule-based Expert Systems: The MYCIN Experiment of the Stanford Heuristic Programming Project, Addison-Wesley Publ. Co, Reading (1984).

28) E. H. Shortliffe and B. G. Buchanan, Mathematical Biosciences 23 (1975) pp. 351-379.

29) see Ref. (27b) pp. 264.

30) see Ref. (19).

31) W. van Melle, Proc. of the Intl. Joint Conf. on Artificial Intelligence 6 (1979) pp. 923-925;
W. van Melle, E. H. Shortliffe and B. G. Buchanan, Machine Intelligence, Infotech State of the Art 9, No. 3 (1981).

32) B. G. Buchanan and E. A. Feigenbaum, Artificial Intelligence 11 (1978) pp. 5-24;
R. K. Lindsay, B. G. Buchanan and E. A. Feigenbaum, The DENTRAL Project, McGraw-Hill (1980) New York.

33) W. J. Clancey, Transfer of rule-based expertise through a tutorial dialogue, Ph. D. Thesis, Computer Science Dept., Stanford University (1979).
W. J. Clancey, in: Methods and Tactics in Cognitive Science, (Eds. W. Kintsch, J. R. Miller and P. G. Polson) Erlbaum, Hillsdale, NJ (1984).

34) W. J. Clancey and R. Letsinger, Proc. of the 7th Intl. Joint Conf. on Artificial Intelligence, Vancouver (1981) pp. 829-836.

35) J. S. Benett, L. Creary, R. Engelmore and R. Melosh, A knowledge-based consultant for structual analysis, Rep.

No. HPP-78-23 (1978), Computer Science Dept., Stanford University.

36) E. H. Shortliffe, A. C. Scott, M. B. Campbell, A. B. van Melle and C. D. Jacobs, ONCOCIN: An Expert System for oncology protocol managment, Proc. of the Intl. Joint Conference on Artificial Intelligence 7 (1981).
C. P. Langlotz and E. H. Shortliffe, Intl. Journal of Man-Machine Studies 19 (1983) 479-496.

37) J. C. Kunz, R. J. Fallat, D. H. McClung, J. J. Osborn, B. A. Votteri, H. P. Nii, J. S. Aitkins, L. M. Fagan and E. A. Feigenbaum, Proc. of Computers in Critical Care and Pulmonary Medicine (1979) pp. 375-379.

38) J. S. Aikins, J. C. Kunz, E. H. Shortliffe, R. J. Fallat, Computers and Biomedical Research 16 (1983) pp. 199-208.

39) D. E. Smith and J. E. Clayton, in: Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project (Eds. B. G. Buchanan and E. H. Shortliffe), Addison-Wesley Publ. Co (1984) pp. 441-452.

40) L. Fagan, Ph. D. Thesis, VM: Representing time-dependent relations in a clinical setting (1980), Computer Science Dept., Stanford University.

41) unpublished, see ref. 27b.

42) J. S. Bennet and D. Goldman, CLOT: A knowledge-based consultant for bleeding disorder, Rep. No. HPP-80-7, Computer Science Dept., Stanford University.

43) J. F. Heiser, R. E. Brooks and J. P. Ballard, Proc. of the 11th Collegium Internationale Neuro-Psychopharmacologicum, Vienna, Austria (1978) pp. 233.

44) J. S. Bennet and C. R. Hollander, Proc. of the 7th Intl. Joint Conference on Artificial Intelligence (Vancouver, B.C.) (1981) pp. 843-845.

45) A. Bonnet, An Expert System for lithographic analysis, Internal working paper, Schlumberger Corp., (1981) Paris, France.

46) B. Mulsant and D. Servant-Schreiber, Computer and Biomedical Research 17 (1984) pp. 71-91.

47) L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Reddy, Computing Surveys 12 (1980) pp. 213-253.

48) D. M. McKeown, Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing, Hartford, Conn. (1977) pp. 795-798.

49) G. Ernst and A. Newell, GPS: A Case Study in Generality and Problem Solving, Academic Press, NY (1969);
N. Nilson, Problem Solving Methods in Artificial Intelligence, McGraw-Hill, NY (1971):
E. E. Sacerdoti, Artif. Intell. 5 (1974) pp. 115-135.

50) R. Balzer, L. Erman, P. London and C. Williams, Proc. of the Natl. Conf. of the American Association for Artificial Intelligence 1, (1980) pp. 108-110.
L. D. Erman, P. E. London and S. F. Fickas, Proc. of the Intl. Joint Conference on Artificial Intelligence 7 (1981) pp. 409-415.

51) N. Goldman, AP3 User's Guide (1978), Information Sciences Institute, Univ. of Southern California, Los Angeles.

52) W. Teitelman, INTERLISP Reference Manual (1978), Xerox Palo Alto Research Center, Palo Alto, California.

53) C. E. Hewitt, Tech. Rep. TR-258 (1972), Artificial Intelligence Laboratory, MIT, Cambridge, Mass.

54) H. P. Nii and E. A. Feigenbaum, in: Pattern-Directed Inference Systems, (eds. D. A. Waterman and F. Hayes-Roth), Academic Press (1978) NY, pp. 483-501.
E. A. Feigenbaum, in: Expert Systems in the Micro Electronic Age (eds. E. Michie) Redwood Burn Ltd. (1979), England, pp. 3-25.

55) E. A. Feigenbaum, R. S. Engelmore and C. K. Johnson, Acta Crystallographica A33 (1977) 13.

56) M. Stefik, Planning with Constraints, Ph. D. Thesis, Computer Science Dept., Stanford University, Stanford, California (1980).

57) B. Hayes-Roth and F. Hayes-Roth, Tech. Rep. R-2366-ONR (1979), The Rand Corporation, Santa Monica, California; B. Hayes-Roth and F. Hayes-Roth, Cognitive Science 3 (1979) 275-310.

58) D. E. Rumelhart, Towards an Interactive Model of Reading, Tech. Rep. 56 (1976), Center for Human Information Processing, University of California, San Diego.

59) R. Balzer, L. D. Erman and C. Williams, HEARSAY-III: A domain-independent base for knowledge-based problem solving, Tech. Rep., USC/Information Sciences Institute, Marina del Rey, California (1980).

60) H. P. Nii and N. Aiello, Proc. 6th Int. Joint Conf. Artificial Intelligence, Tokyo (1979) pp. 645-655.

61) A. R. Hanson and E. M. Riseman, VISIONS: A computer system for interpreting scenes, Academic Press, NY (1978) pp. 303-333.

62) E. M. Solovay and E. M Riseman, Proc. 5th Intl. Joint Conf. Artificial Intelligence, Cambridge, Mass. (1977) pp. 801-811.

63) M. D. Levine, in: Computer Vision Systems (Eds. A. Hanson and E. Riseman) Academic Press, NY (1978) pp. 335-352.

64) W. C. Mann, 17th Ann. Meeting Assoc. Computational Linguistics, La Jolla, California (1979).

65a) R. O. Duda, J. Gaschnig, P. E. Hart, K. Konolige, R. Reboh, P. Barrett and J. Slocum, Development of the PROSPECTOR consultation system for mineral exploration, SRI Projects 5821 and 6415, SRI International, Inc., Menlo Park, Calif. (1978).

65b) R. Duda. J. Gaschnig. P. E. Hart, in: Expert Systems in the Micro-Electronic Age, Edinburgh University Press (1979) pp. 153-167.
R. Duda and J. G. Gaschnig, BYTE 6 (1981) pp. 238-281.

66) H. Pople, Proc. of the Intl. Joint Conf. on Artificial Intelligence 5 (1977) pp. 1030-1037.

67) G. G. Hendrix, in: Associative Networks - The Representation and Use of Knowledge in Computers (Eds. N.V.Findler) Academic Press, N.Y. (1979) pp. 51-92.

68) L. A. Zadeh, "Fuzzy Sets", Information and Control 8 (1965) pp. 338-353.
L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility", Fuzzy Sets and Systems 1 (1978) pp. 3-28.

69) G. G. Hendrix, SIGART Newsletter 61 (1977) pp. 25-26.
G. G. Hendrix, The LIFER manual: A Guide to building practical natural language interfaces, Tech. Note 138, AI-Center, SRI International, Inc., Menlo Park, California.

70) R. Reboh, Ph. D. Thesis, Knowledge Engineering Techniques and Tools for Expert Systems, Linkoping University, Sweden (1981).

71) T. Johnson, The Commercial Application of Expert Systems Technology, Ovum Ltd., London (1985)
B. Humpert, Artificial Intelligence: Impact on Pure and Applied Science, HASLER F+S AI/113 (1985), to be published in Helv. Phys. Acta.

72) See for example in Ref. [15], pp. 400, and Ref. [28]

73) D. B. Lenat et al. AI Magazine 3 (1982) 17

74) M. Schindler, Electronic Design, August 1984, pp. 106-146; ibid, January 1985, pp. 113-134

FIGURE CAPTIONS

Fig. 3.1 : Forward and Backward chaining in production systems.

Fig. 5.1 : The program structure of the MYCIN-system.

Fig. 5.2 : A MYCIN production rule.

Fig. 5.3 : A MYCIN goal tree.

Fig. 5.4 : The search tree of a toy model for MYCIN.

Fig. 5.5 : The interrelation of MYCIN-type expert systems.

Fig. 6.1 : The blackboard program structure of the HEARSAY-II system.

Fig. 6.2 : The levels and knowledge sources in HEARSAY-II.

Fig. 7.1 : The program structure of the PROSPECTOR-system.

Fig. 7.2 : The size of the knowledge base of five PROSPECTOR models.

Fig. 7.3 : Examples of taxonomies in the PROSPECTOR system.

Fig. 7.4 : Semantic network representation of: "a rhyolite plug is present".

Fig. 7.5 : A sample inference network from a PROSPECTOR model.

Fig. 8.1 : List of expert-systems and tools (partially from Ref. 74).

Fig. 8.2 : Application areas of expert-systems (adapted from Ref. 71).

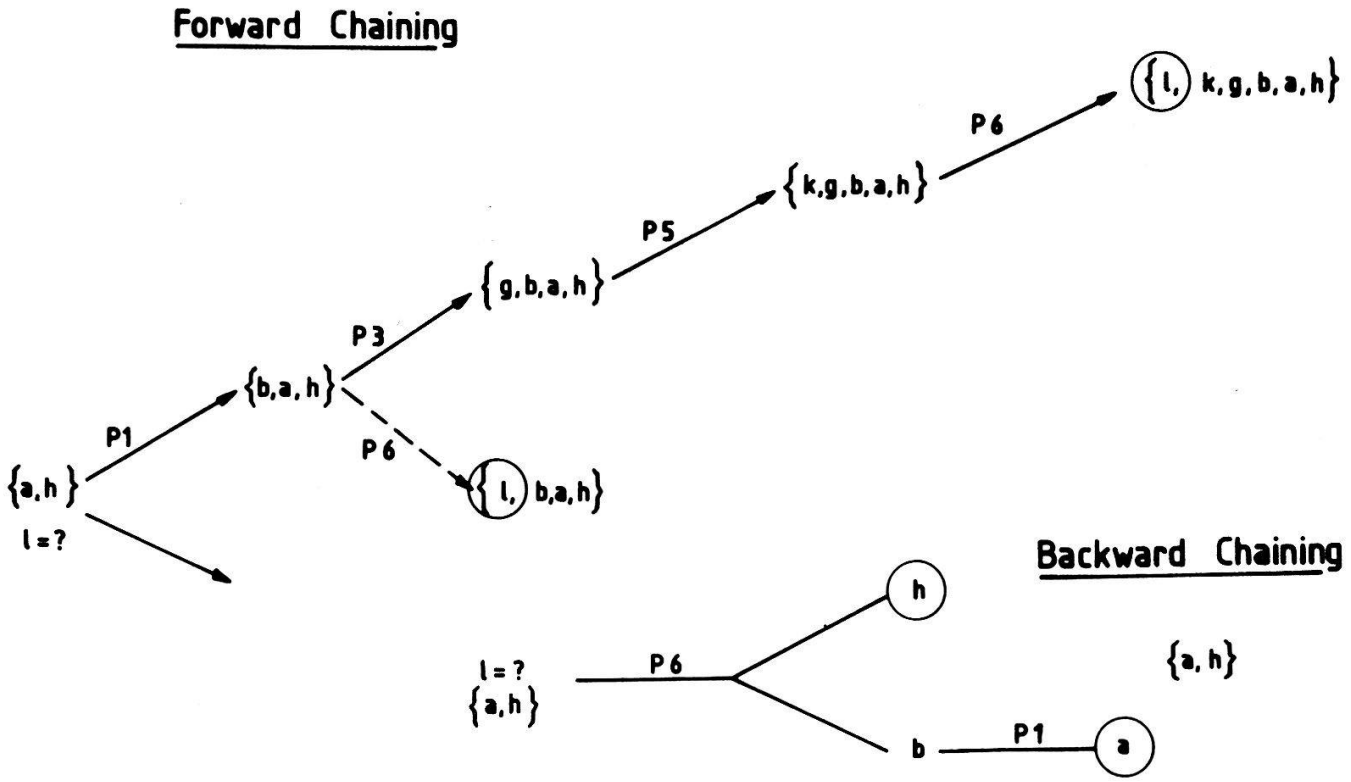

Forward Chaining
{a,h}
l=?
P1
{b,a,h}
P3
{g,b,a,h}
P5
{k,g,b,a,h}
P6
{l, k,g,b,a,h}
P6
{l, b,a,h}
Backward Chaining
l=?
{a,h}
P6
h
b
P1
a
{a,h}

Fig. 3.1

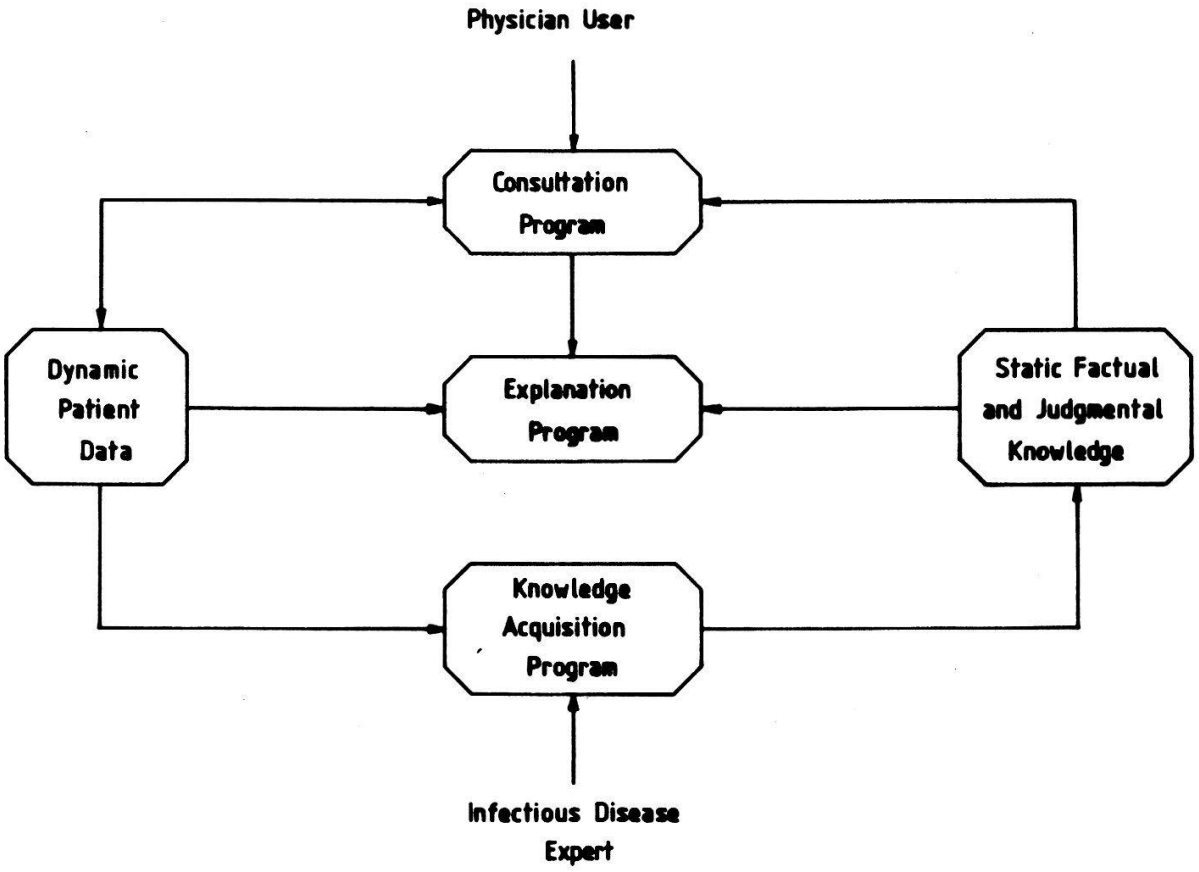

Physician User
Consultation Program
Dynamic Patient Data
Explanation Program
Static Factual and Judgmental Knowledge
Knowledge Acquisition Program
Infectious Disease Expert

Fig. 5.1

Rule 050

```
PREMISE:  (AND (SAME CNTXT INFECT PRIMARY-BACTEREMIA)
               (MEMBF CNTXT SITE STERILESITES)
               (SAME CNTXT PORTAL GI) )
ACTION:   (CONCLUDE CNTXT IDENT BACTEROIDES TALLY .7)
```

MYCIN's English translation:

IF 1) the infection is primary-bacteremia, AND
2) the site of the culture is one of the sterile sites, AND
3) the suspected portal of entry of the organism is the gastrointestinal tract,

THEN there is suggestive evidence (.7) that the identity of the organism is bacteroides.

Fig. 5.2

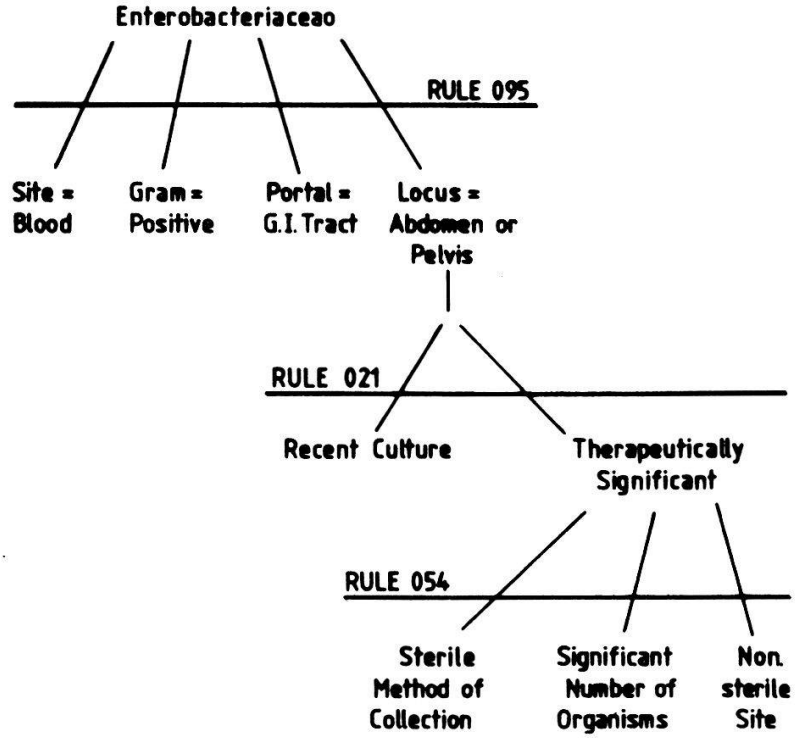


Fig. 5.3

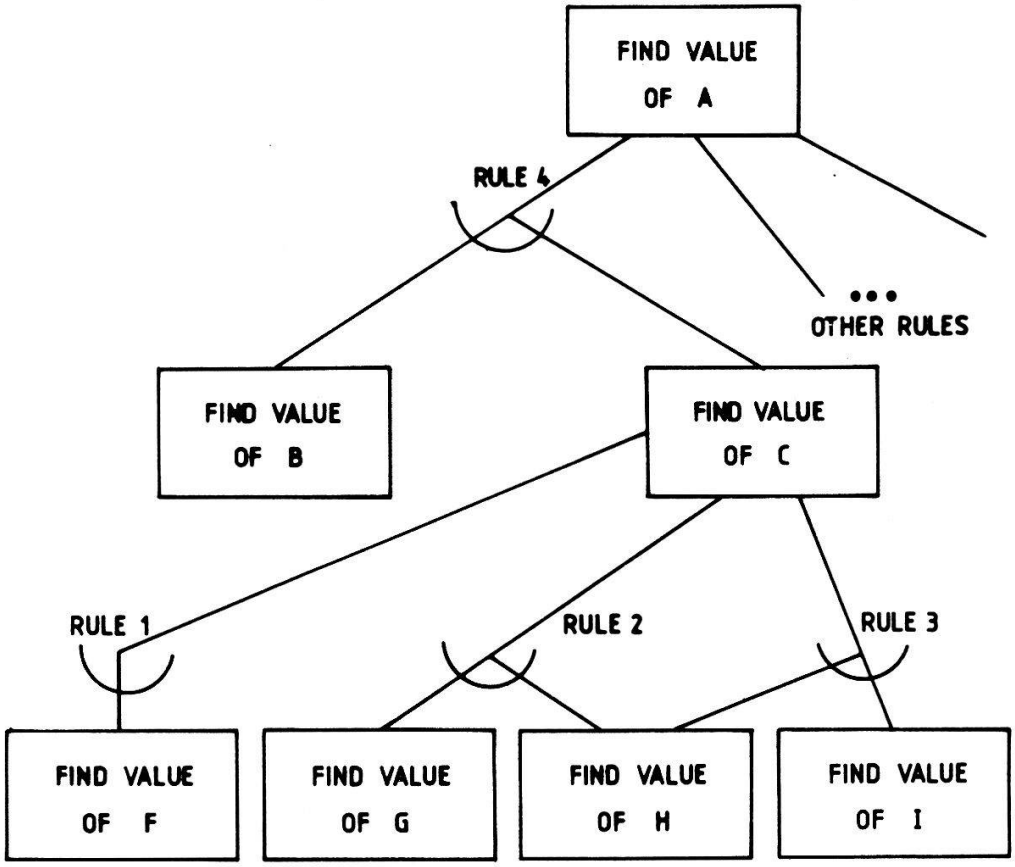

FIND VALUE OF A
RULE 4
OTHER RULES
FIND VALUE OF B
FIND VALUE OF C
RULE 1
RULE 2
RULE 3
FIND VALUE OF F
FIND VALUE OF G
FIND VALUE OF H
FIND VALUE OF I

Fig. 5.4

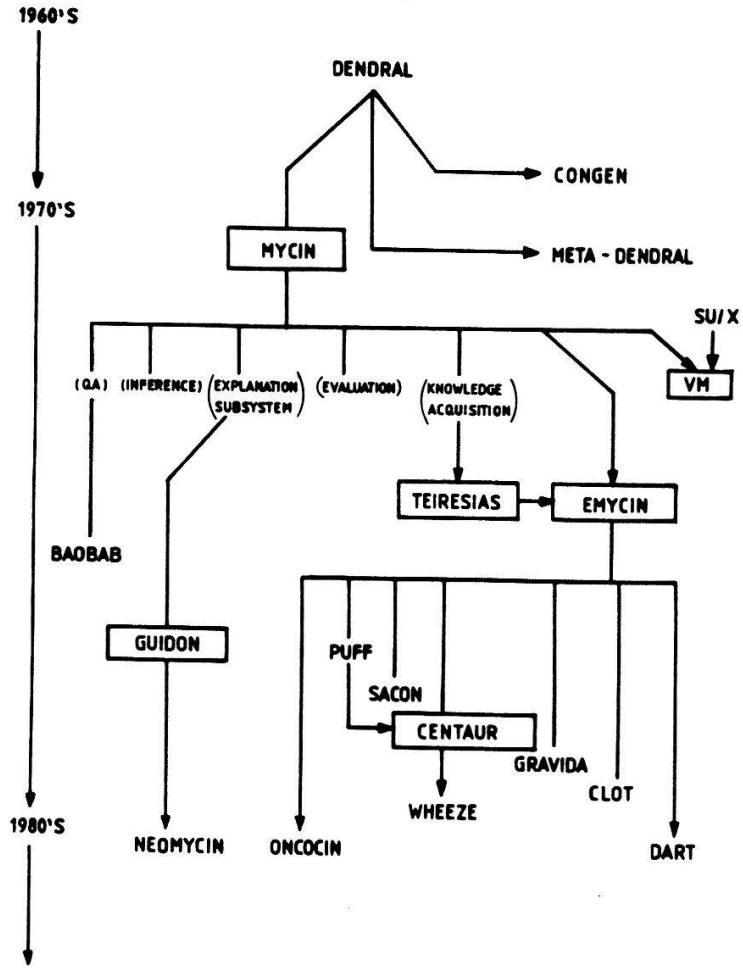

1960'S
1970'S
1980'S
DENDRAL
CONGEN
META - DENDRAL
MYCIN
SU/X
VM
(QA)
(INFERENCE)
EXPLANATION SUBSYSTEM
(EVALUATION)
KNOWLEDGE ACQUISITION
TEIRESIAS
EMYCIN
BAOBAB
GUIDON
PUFF
SACON
CENTAUR
GRAVIDA
CLOT
WHEEZE
NEOMYCIN
ONCOCIN
DART

Fig. 5.5

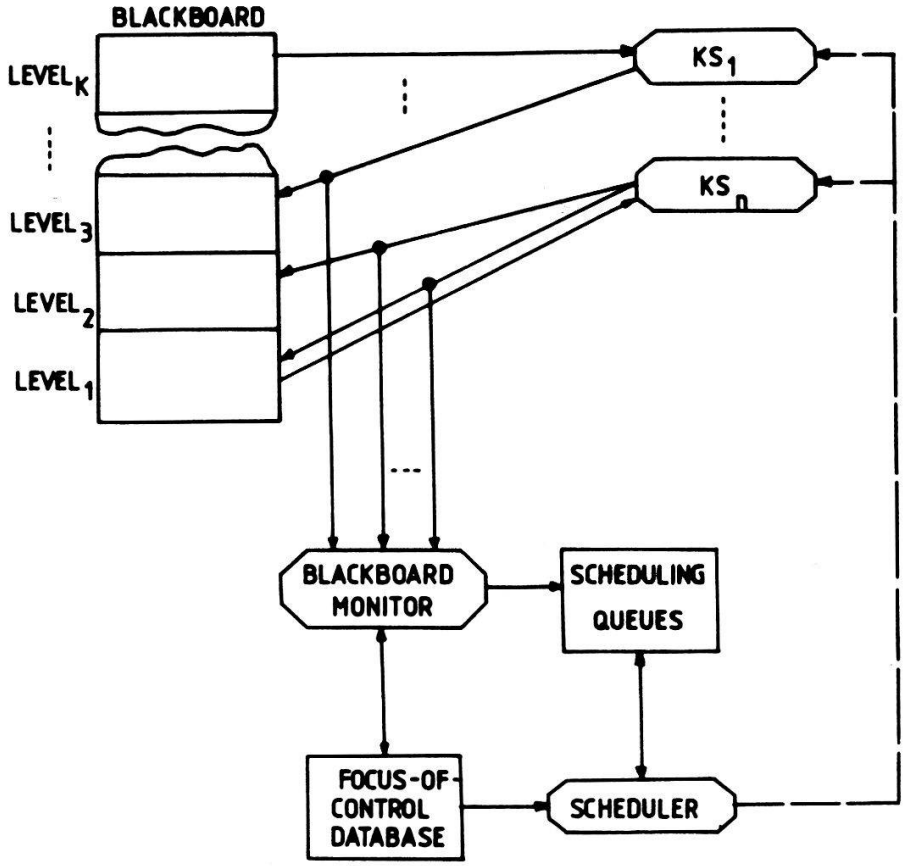

BLACKBOARD
LEVEL K
LEVEL 3
LEVEL 2
LEVEL 1
KS 1
KS n
BLACKBOARD MONITOR
SCHEDULING QUEUES
FOCUS-OF-CONTROL DATABASE
SCHEDULER

Fig. 6.1

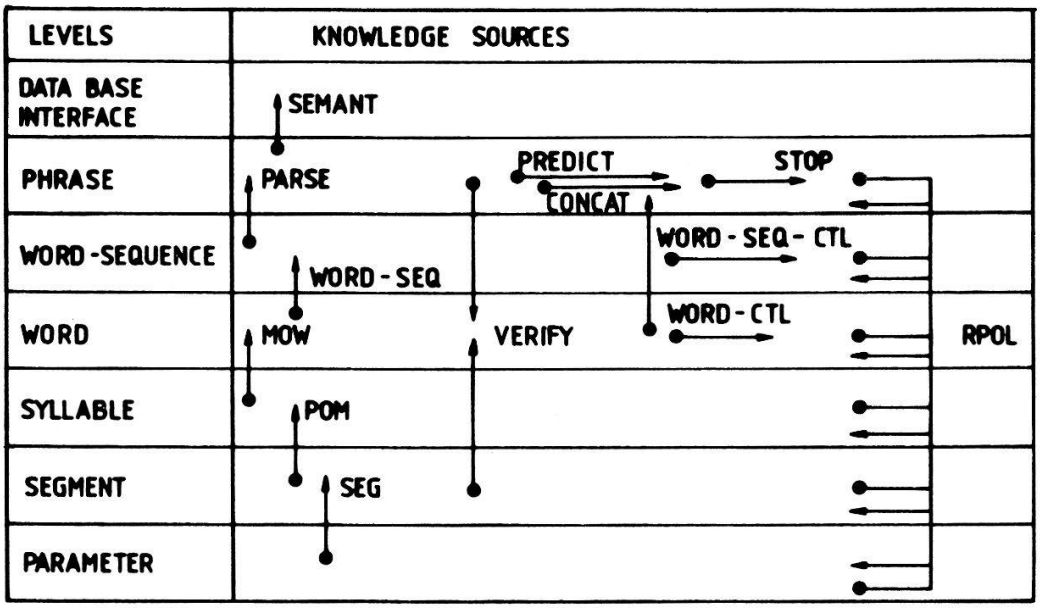

LEVELS
KNOWLEDGE SOURCES
DATA BASE INTERFACE
SEMANT
PHRASE
PARSE
PREDICT
STOP
CONCAT
WORD-SEQUENCE
WORD-SEQ-CTL
WORD-SEQ
WORD
MOW
VERIFY
WORD-CTL
RPOL
SYLLABLE
POM
SEGMENT
SEG
PARAMETER

Fig. 6.2

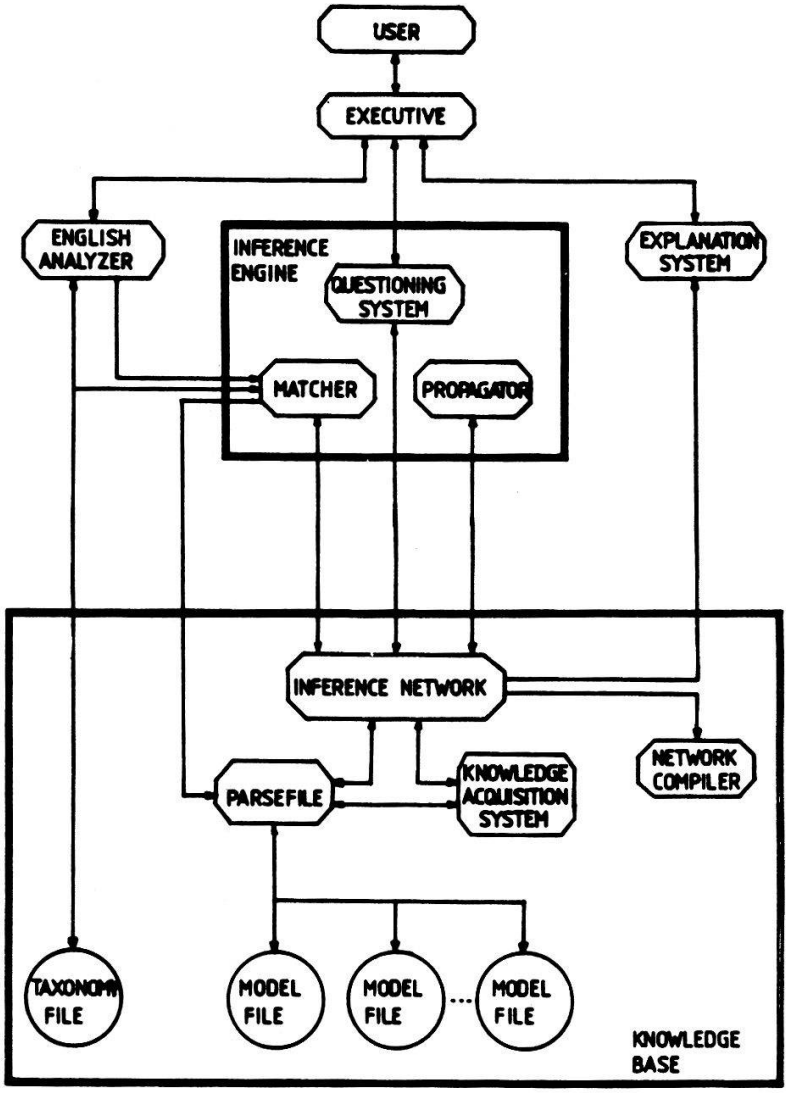


Fig. 7.1

Model	Number of assertions	Number of rules
Koroko-type massive sulfide	39	34
Mississippi-Valley-type lead/sinc	28	20
Type A porphyry copper	187	91
Komatiitic nickel sulfide	75	49
Roll-front sandstone uranium	212	133
Total	541	327

Fig. 7.2

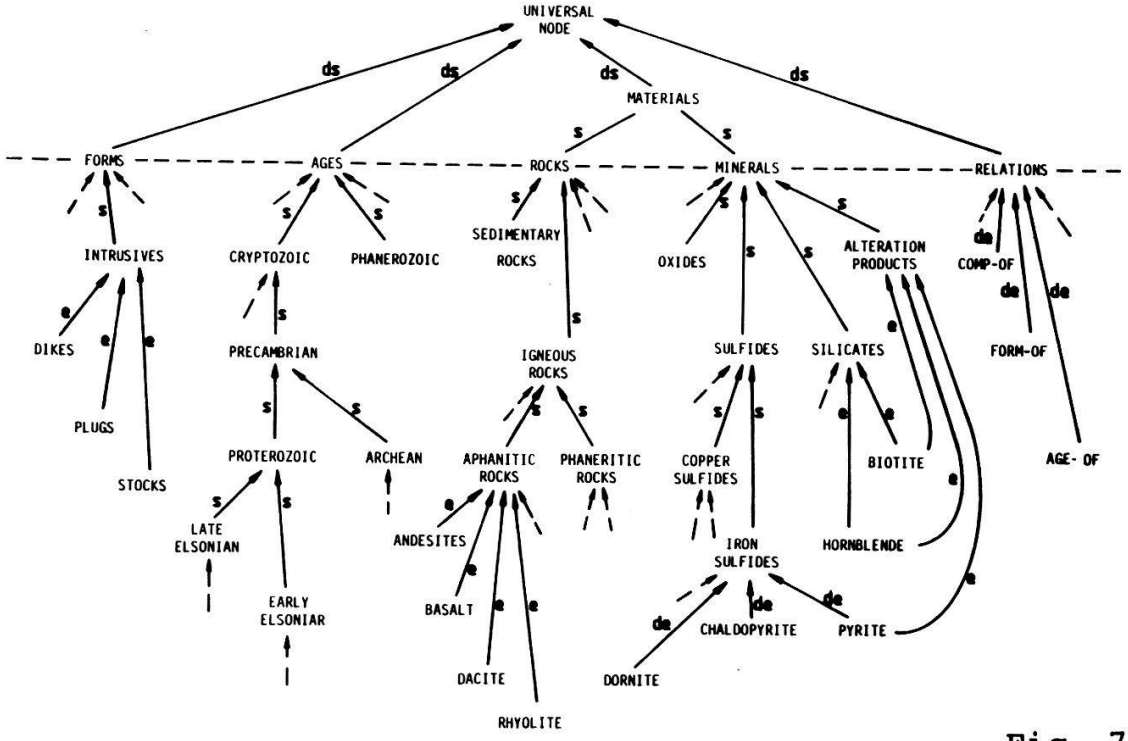
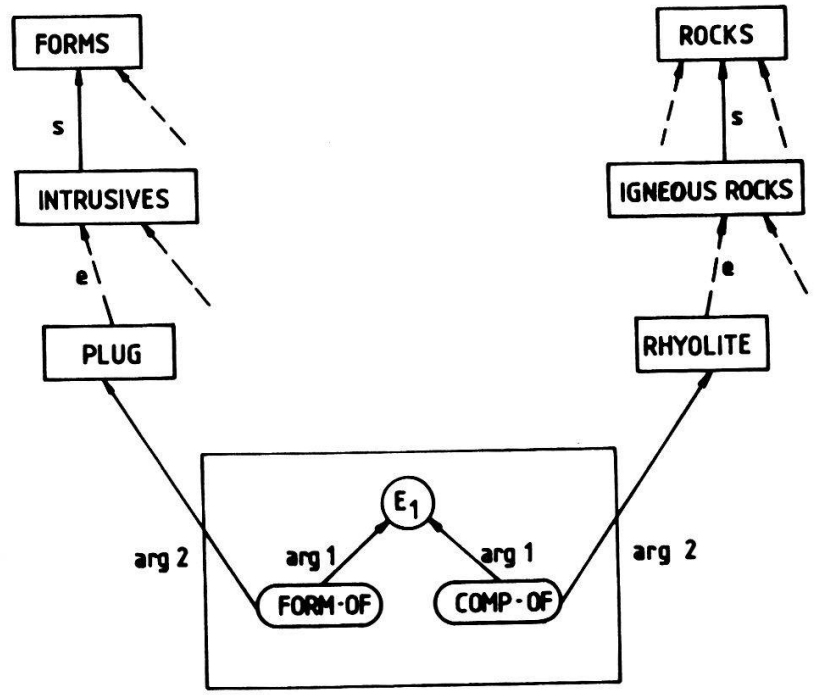


Fig. 7.3



"a rhyolite plug is present"

Fig. 7.4

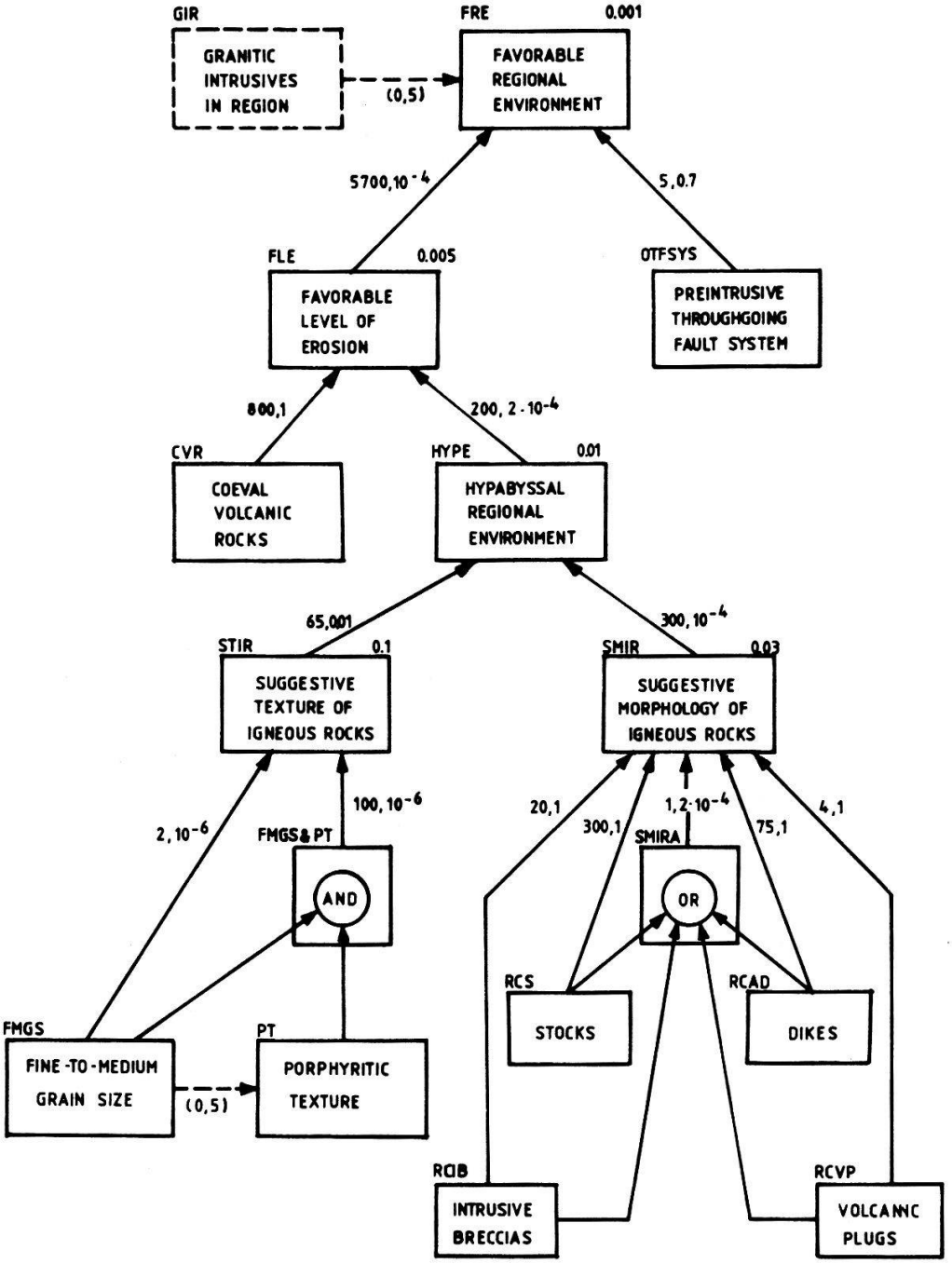

GIR
GRANITIC INTRUSIVES IN REGION
(0,5)
FRE
0.001
FAVORABLE REGIONAL ENVIRONMENT
5700,10⁻⁴
5,0.7
FLE
0.005
FAVORABLE LEVEL OF EROSION
OTFSYS
PREINTRUSIVE THROUGHGOING FAULT SYSTEM
800,1
200, 2·10⁻⁴
CVR
COEVAL VOLCANIC ROCKS
HYPE
0.01
HYPABYSSAL REGIONAL ENVIRONMENT
65,001
300, 10⁻⁴
STIR
0.1
SUGGESTIVE TEXTURE OF IGNEOUS ROCKS
SMIR
0.03
SUGGESTIVE MORPHOLOGY OF IGNEOUS ROCKS
2, 10⁻⁶
100, 10⁻⁶
FMGS& PT
AND
20,1
300,1
1,2·10⁻⁴
SMIRA
75,1
4,1
OR
RCS
STOCKS
RCAD
DIKES
FMGS
FINE-TO-MEDIUM GRAIN SIZE
(0,5)
PT
PORPHYRITIC TEXTURE
RCIB
INTRUSIVE BRECCIAS
RCVP
VOLCANIC PLUGS

Fig. 7.5

ABEL
Expert system for diagnosing electrolytic disorders

ACE
Expert system that analyzes trouble reports for telephone cables.

AGE
Attempt to generalize-helps design expert systems like PUFF.

AIPS
Advanced information presentation system-expert system for graphical objects.

AL/X
Assists in assembling diagnostic expert systems, based on Prospector.

AM
Expert system that assists in forming mathematical concepts.

APE
Expert System for automatic programming in LISP.

ARBY
Construction aid for expert systems in electronic systems-analysis.

ARGOS
General system simulating the decision taking of a robot.

ART
Expert system for building different types of expert systems.

AURA
Automated reasoning assistant-helps with logic and software disign.

BACON-3
Experimental system concerning physics, the laws by Kepler, Coulomb and Ohm.

BAOBAB
Rule manipulation system.

BETA
Battlefield exploitation and target acquisition-an experimental system.

CAA
Casual arhythmia analysis-expert system for analyzing electrocardiograms.

CADUCEUS
Expert diagnostic system for internal medicine (undergoing clinical trials).

CALLISTO
Experimental expert system for the management of large projects.

CASNET
Casual network- associates glaucoma treatment with diagnostic hypotheses.

CATS-1
Operational expert system for troubleshooting diesel-electric locomotives.

CHI
Experimental expert system for automatic software development.

CMUDA
CMU design automation-comprises heuristic and algorithmic VLSI tools.

CRIB
Expert system to train field engineers in resolving computer malfunctions.

CRITTER
Expert system to critique VLSI designs interactively.

CRYSALIS
Expert system for data analysis related to protein crystallography.

DAA
Design automation assistant-Al-based VLSI design segment of CMUDA.

DART
Experimental expert system for field diagnosis of computer system faults.

DELTA
An earlier term for CATS-1.

DEMETER
Design methodology and environment-for system design above register level.

DENDRAL
Expert system that determines molecular structure from mass spectrograms.

DIPMETER
Expert system that analyzes oil well data.

DOC
Prolog-based expert sys-

Fig. 8.1

tem for computer field service.

DPL
Description language for VLSI design.

DWIM
Do what I mean-context-based error correction in LISP systems.

EDD
Expert data base designer-a Prolog-based expert system.

EL
Expert system for circuit analysis.

ELAS
Expert system for analysis of oil-well data.

EMUCS
Algorithmic part of CMUDA. Used for data-path synthesis.

EMYCIN
Inference system of Mycin, used to design expert systems.

EPM
Extended program model-software representation for IPE.

EURISKO
Self-learning expert system for VLSI design.

EXPERT
Basic inference system used in medical applications.

EXPERT-EASE
Tool to design expert systems on a personal computer.

FRL
Knowledge representation language for frame manipulation.

GA1
Expert system for chemical data analysis.

GARI
Expert system for production line.

GEM
Interface management system used in Steamer.

GENESIS
Expert system for genetic engineering.

GEN-X
Inference engine derived from CATS-1.

GLISP
Programming language that supports objects and their behavior.

GUIDON
Experimental computer-aided instruction system for medical & technical applications.

HAIL-1
Expert system that configures printed circuit boards.

HEADMED
Expert system for pharmacological advice.

HEARSAY
Expert system to assist in software design.

HYDRO
Consultation system for solving water resource problems.

IDT
Expert system for diagnosing computer faults.

INFORM
Knowledge-based software environment for rapid prototyping.

INTERNIST
Expert system for internal medicine-alternative name for Caduceus.

IPE
Intelligent program editor-expert system that analyzes software.

ISIS
Experimental expert system for job-shop scheduling.

KAS
Experimental knowledge acquisition system using rule networks.

KBPA
Knowledge-based program automation.

KBSA
Knowledge-based software automation-a life-cycle support system.

KBVLSI
Experimental expert system for VLSI design.

KEE
Tool for assembling knowledge bases for expert systems.

KEPE
Knowledge representations system.

KL-TWO
Knowledge engineering tool for expert systems based on KL-One.

KMS
Assists in building expert systems for medical diagnosis.

KRL
Knowledge representation language-used in framebased systems.

KRYPTON
Knowledge representation system using frame-based and logic-based terms.

KS-300
Basic inference system for industrial diagnostic applications.

LIBRA
Efficiency-analysis component of PSI software automation project.

LITHO
Expert system in geology.

LOOPS
Object-based knowledge representation system, for VLSI and other expert systems.

MACPITTS
Hardware specification methodology for algorithmic VLSI design.

MACSYMA
Knowledge-based system for symbolic mathematical manipulations.

MARS
Multiple abstraction rule-based simultor-used in Palladio.

MAXWELL
Knowledge base development system written in Prolog.

MDX
Expert system for medical diagnosis.

MECHO
Expert system to simulate mechanics.

METADENDRAL
Helps formulate rules regarding fragmented molecules from mass spectrometer data.

METALOG
Logic language.

MICON
Expert system for designing single board computers.

MOLGEN
Expert system for planning bio-engineering experiments involving DNA.

MRS
Metalevel representation system-used for knowledge bases and problem solving.

MYCIN
Operational expert system for diagnosing infectious diseases.

NASL
Expert system for electronics simulations.

NETL
Knowledge representation language for frame-based systems.

NEWTON
Experimental expert system that analyzes the effects of gravity.

NIKL
Part of KL-Two that handles structured predicates.

NOAH
Expert system for planning robotics projects.

NUDGE
Expert system for temporary employment.

OBJTALK
Implementation language for Inform project.

ONCOCIN
Expert system that guides physicians in administering chemotherapy.

OPS4, OPS5
Knowledge representation and inference system developed for R1.

PA
Programmer's apprentice-experimental programming consultant.

PALLADIO
Knowledge-based VLSI design environment based on Loops.

PART
Heuristic program for the solution of arithmetic exercises.

PEACE
System for the electricity simulations.

PECOS
Program synthesis component of PSI software

Fig. 8.1 (con't)

automation system.

PENNI
Part of KL-Two that works with propositions.

PIP
Expert system for medical diagnosis.

POLITICS
Expert system for natural language comprehension.

PONTIUS
Expert system for flight instructions.

PRL
Program reference language-used in IPE project.

PROLOG
Programming language in logic.

PROSPECTOR
Expert system that evaluates sites for their mineral deposits.

PROUST
Experimental expert system that analyzes Pascal programs.

PSI
Expert system that converts English language specifications to simple programs.

PSN
Procedural semantic network-knowledge base using classes, objects, and relations.

PTRANS
Rule-based expert system for manufacturing management.

PUFF
Operational expert system for diagnosing lung diseases.

QUAL
Expert system that explains an electrical circuit's behavior from its description.

R1
Original name for XCON which configures VAX systems.

RAFFIES
Precompiles the knowledge for CRIB.

REDESIGN
Expert system for VLSI design that emphasizes modifications.

RESEDA
Expert system for history and biographies.

RITA
Language for computer aided conception.

REX
Regression expert-frame, based expert system for statistical analysis.

ROGET
Experimental system that adds knowledge engineering expertise to Emycin.

ROSIE
Basic inference system used in several applications.

RUP
Reasoning utility package-predecessor of Penni.

RX
Expert system for evaluating statistical data from chronically ill patients.

SACON
Operational expert system for structural analysis.

SAFE
Experimental software automation system.

SAGE
Basic inference system used in several applications.

SAM
Expert system for the treatment of hypertension.

SCHEMA
Expert system for VLSI design-still in early development stage.

SCHOLAR
Computer-aided instruction system for coaching games.

SECS
Operational expert system that helps chemists plan organic synthesis.

SIMMIAS
Expert system for geological analyses.

SMALLTALK
Language and environment for graphics and object based programming.

SMP
Symbolic manipulation program-solves differential, integral, transcendental equations.

SNARK
Language in predicate logics.

SNIFFER
Experimental expert system that discovers software bugs.

SOPHIE
Computer-aided instruction for troubleshooting electronics.

SPEAR
Expert system for field analysis of error logs-under development.

SPHINX
Project system in medicine.

SRL
Schema representation language-used for knowledge description.

STEAMER
Expert system that simulates steam plants for Navy training.

SU/X
Expert system for signal processing.

SYN
Expert system for circuit synthesis.

SYSTEM-1
Knowledge engineering tool for expert systems-nearly operational.

TALIB
Expert system that synthesizers layouts for NMOS cells.

TEIRESIAS
Guides knowledge and rule acquisition for expert systems.

TIMM
The intelligent machine model-implements expert systems in Fortran.

TROPIC
Expert system for computer aided conception in architecture.

TRUCKIN
An expert system game to train students in the use of Loops.

TUNER
Pascal-based expert system for adjusting signal-processing systems.

UNITS
Knowledge representation system used with AGE to build Molgen.

VEXED
VKSI expert editor-experimental self-teaching IC design system.

VM
Ventilator management-monitors patients and suggests respiratory therapy.

WAVES
Expert system for analyzing seismic data for the oil industry, uses KS-300.

XCON
Current version of R1 expert system for configuration of VAX and PDF-11 systems.

XPLAIN
Expert system framework that accounts for its reasoning.

XSEL
Extension to XCON that supplies salesmen with floor plans.

Fig. 8.1 (con't)

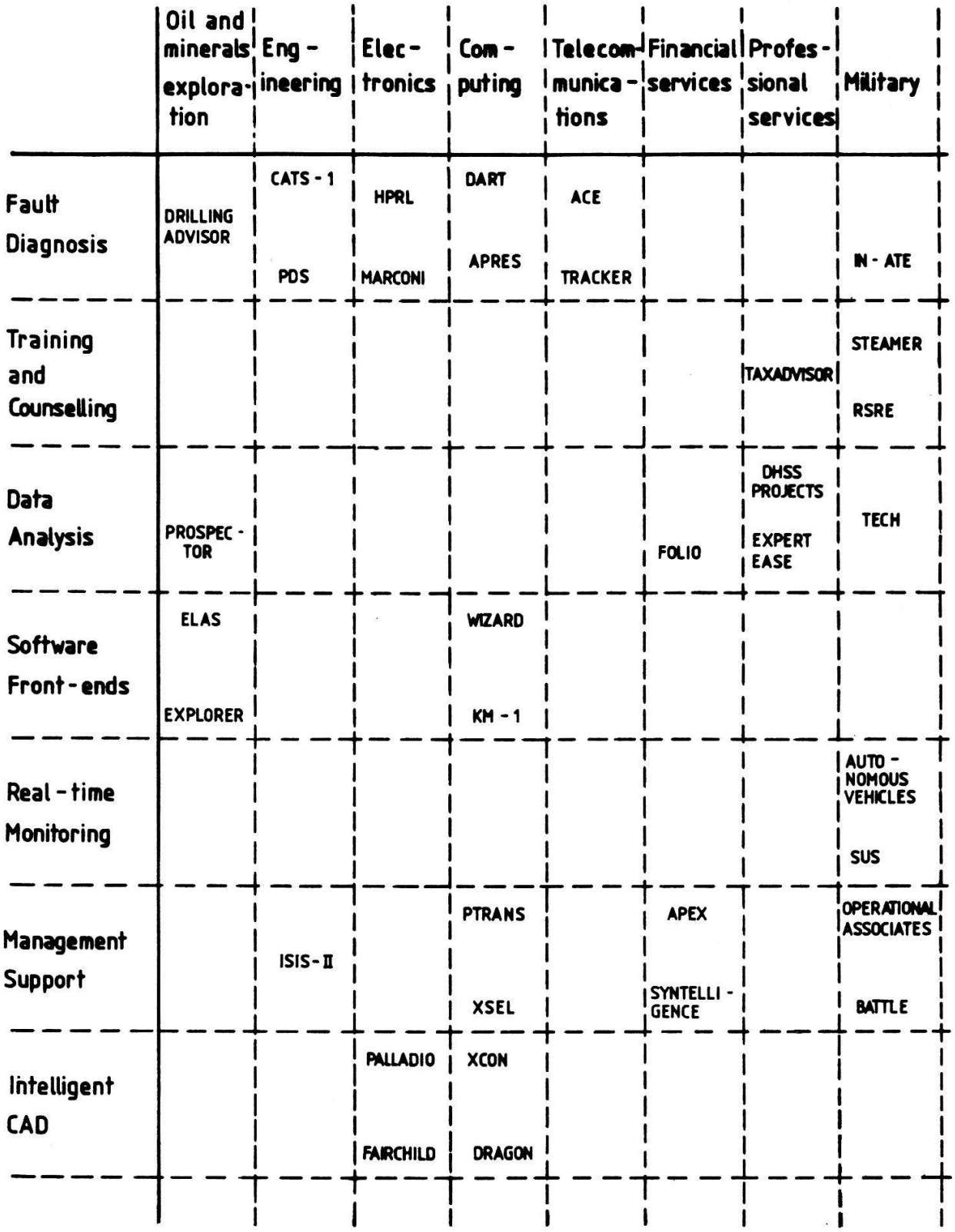

Oil and minerals exploration
Engineering
Electronics
Computing
Telecommunications
Financial services
Professional services
Military
Fault Diagnosis
DRILLING ADVISOR
CATS-1
PDS
HPRL
MARCONI
DART
APRES
ACE
TRACKER
IN-ATE
Training and Counselling
TAXADVISOR
STEAMER
RSRE
Data Analysis
PROSPECTOR
FOLIO
DHSS PROJECTS
EXPERT EASE
TECH
Software Front-ends
ELAS
EXPLORER
WIZARD
KM-1
Real-time Monitoring
AUTONOMOUS VEHICLES
SUS
Management Support
ISIS-II
PTRANS
XSEL
APEX
SYNTELLIGENCE
OPERATIONAL ASSOCIATES
BATTLE
Intelligent CAD
PALLADIO
FAIRCHILD
XCON
DRAGON

Fig. 8.2