Proceedings of the first
European Workshop on

# KNOWLEDGE ACQUISITION FOR KNOWLEDGE - BASED SYSTEMS

Co - Sponsored by the
Institution of Electrical Engineers



2nd - 3rd September 1987
Reading University

# Proceedings of the first European Workshop on

# KNOWLEDGE ACQUISITION FOR KNOWLEDGE - BASED SYSTEMS

Co - Sponsored by the

Institution of Electrical Engineers

**Co-Chairmen:**


Tom Addis
Department of Computer Science
University of Reading
Whiteknights, PO Box 220
Reading RG6 2AX
(Tom Addis@reading.ac.uk)

John Boose
Boeing Artificial Intelligence Centre
Boeing Computer Services M/S 7A-03
PO Box 24346
Seattle, Washington, USA 98124
(john@boeing.com)

Brian Gaines
Department of Computer Science
University of Calgary, Calgary,
Alberta, Canada T2N 1N4
(gaines@calgary.cdn)


2nd - 3rd September 1987

Reading University

# CONTENTS


**COLLOQUIUM**

A1 *A Framework for Knowledge Elicitation*
T R Addis

A2 *Mediating representations in Knowledge Elicitation*
Nancy Johnson

A3 *Knowledge Acquisition for Expert Systems*
Brian R Gaines

A4 *Knowledge Acquisition and Machine Learning - The Issue of Modelling*
K Morik

A5 *Machine Learning and Explanations*
Y Kodratoff

A6 *AQUINAS: A Knowledge Acquisition Workbench for Building Knowledge-Based Systems*
J Boose, J M Bradshaw

**THEORY**

B1 *Knowledge Acquisition as modelling expertise: the KADS Methodology*
J Breuker, B Wielinga

B2 *How do Experts Acquire Expertise?*
Brian R Gaines

**PRACTICE**

C1 *A Knowledge Acquisition extension to Notecards*
Brian R Gaines, M Sharp

C2 *Knowledge Acquisition for Real-Time Knowledge Based Systems*
B Clarke

C3 *Knowledge Acquisition for the Advice System*
E S Cordingley

C4 *Knowledge Acquisition and Nuclear Advisors*
B McHugh

C5 *Experience of Knowledge Acquisition for Expert Systems in Construction*
G Trimble, C N Cooper

C6 *Knowledge Acquisition: A Survey and British Telecom Experience*
M Welbank

**TECHNIQUES**

D1 *On Structuring Knowledge for Incremental Knowledge Acquisition*
Marc Linster

D2 *Teaching Knowledge Engineers the Human Side of Knowledge Engineering*
Jill Bell

D3 *A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1*
A M Burton, N R Shadbolt, A P Hedgecock, G Rugg

D4 *POMESS: A People Orientated Methodology for Expert System Specification*
D Diaper

D5 *Knowledge Acquisition for Method and Tool Selection*
H L Hausen, H J Neusser

D6 *Knowledge Acquisition for Small Scale Expert Systems from Consulting Experts: A Field Study Approach*
P Watkins, D E O'Leary

D7 *Role of Psychological Techniques and Intermediate Representations in Knowledge Elicitation*
R M Young, J Gammack

D8 *A Decompilation Approach to Knowledge Acquisition: Using Concurrent Expert Reports as the Basis for Automated Knowledge Modeling*
O Laske

**TOOLS AND MODELLING**

E1 *The Role of Ontology in Concept Acquisition for Knowledge-Based Systems*
I Monarch, S Nirenburg

E2 *The KADS System*
A Anjewierden

E3 *KRIMB: An Intelligent Knowledge Acquisition and Representation Program for Interactive Model Building*
L A Cox, R Blumenthal

E4 *A Methodology for Integrating Knowledge Acquisition Techniques*
J Diederich, Marc Linster, I Ruhmann, T Uthmann

E5 *KET, A Knowledge Encoding Tool*
L Esfahani, F N Teskey

E6 *A Naturalistic Methodology for Knowledge Engineering*
R J Hardiman

E7 *An Interactive Strategic Planning System*
R V Williams, J R Thomas

E8 *Explanation by Context Recognition*
S Thorpe

**INDUCTION**

F1 *An Experimental Protocol for the Acquisition of Examples for Learning*
J Blythe, P Corsi, D Needham

F2 *Automatic Induction of Rules from Examples: A Critical Analysis of the ID3 Family of Rule Induction Systems*
M Bramer

F3 *ITRULE: An Information Theoretic Rule-Induction Algorithm*
R M F Goodman, P Smyth

F4 *Student Modelling: Learning a Function from examples while knowing a Close Function*
O Maler

F5 *Graph Induction*
G D Oosthuizen

**An index to authors is given at the end of these proceedings.**

# A FRAMEWORK FOR KNOWLEDGE ELICITATION

T. R. Addis

Knowledge Systems Group
Department of Computer Science
University of Reading
Whiteknights
Reading, Berkshire

*Abstract*

*Knowledge elicitation is an activity that has aspects of both design and theory formation. Knowledge, as it relates to machines, is analysed and a theoretical framework for the evolution of science is examined with respect to the process of knowledge elicitation. Design is described in terms of a series of abstract machines which are each defined in terms of a symbol set and an implicit grammar. An abstract machine supports a deductive model that will simulate the behaviour of an expert. This model is used to devise "tests" that are assessed by the expert in order to modify the model and consequently create a better model. It is shown that expertise which involves reasoning other than deduction can be simulated by at least a two level model. Two distinct modes of development of a model are identified. These are progressive and non-progressive. Associated with these two modes are the two types of abduction required for the creation of new hypotheses.*

## 1. INTRODUCTION

Knowledge elicitation is the process of creating a representation of an expert's (or experts') competence in a field of activity. The expert's field of activity is often referred to as the *task domain* and is the total set of tasks in which the expert is expected to be accomplished. The process of *knowledge elicitation* is motivated by the need to represent the knowledge which describes an expert's performance in a form that is suitable for simulation by a knowledge-based system. The knowledge-based system is then expected to predict accurately the expert's behaviour within certain restricted situations. Such representations are not neutral in that they depend upon the kind of knowledge-based system and the principles upon which it is constructed. The question arises as to whether it is possible to provide a framework for knowledge elicitation that is based upon general principles and will be applicable to a wide range of techniques; is it possible, for example, to propose a theory of knowledge elicitation?

It is clear from those who practice knowledge elicitation and who may be referred to as *knowledge engineers* or *systems analysts* that the process of knowledge elicitation is similar in some respects to the design of a behavioural simulator and in other respects to the formation of a competence theory. The practical result of knowledge elicitation is usually a model of an expert's behaviour and this model can be considered as a theory of behaviour that matches the observations obtained from interviews or the records of an expert's activities. However, these observations are not always simple. The observations can be statements that generalise over a range of conceivable activities and are themselves models or theories. The model or theory should provide a description which encompasses all the statements made but in practice may only cover a subset. What is finally represented in the knowledge-based system is often a matter of judgement which does not always depend upon the experts since the issues usually involve engineering decisions. These decisions belong to an abstract machine that is defined in terms of the particular knowledge representation scheme adopted by the knowledge engineer.

Related to each abstract machine are symbols and an implicit grammar which together define the range of possible configurations of the machine. It is the abstract machine that limits the kind of knowledge which may be represented and this limitation will influence the style of interview that takes place between the knowledge engineer and the expert. There are many collections of symbols that form a coherent set and are designed to highlight different types of knowledge (Sowa, 1984, Addis, 1985). Each of these symbol sets is concerned with the first steps towards a specific design of a knowledge-base.

Design of a complex artifact normally progresses through a series of stages where each stage represents a different abstract machine. Each abstract machine emphasises different aspects of the design problem and each stage takes the design a step nearer to the construction of an artifact. An example of this design approach is found in the creation of a VLSI wafer for micro-processing. In this example there can be as many as nine different design stages. The final three are electronic circuit design, stick diagrams and area layout. Electronic circuit design has symbols and an implicit grammar that are specifically orientated towards capturing the electronic knowledge for a particular circuit. The area layout encompasses the knowledge of the shapes and the physical material of the layers that form the chip. The stick diagram represents the knowledge of how the electronic components are created from the physical conjunction of the layers (Begg, 1984). Similar shifts of design concern and the associated change in representation are found in data-base design. In this case the stages are the conceptual model, the logical model and the physical schema (Addis, 1985). It is in this sense that knowledge elicitation may be considered as an aspect of the design process.

On the other-hand, an expert's perception of a task domain may be determined by the part which he plays within the task domain, the functions he carries out and the interactions he experiences. In this way the same task domain could be described very differently by two individuals who, due to different objectives and requirements, see different facets of the domain as being important. Thus any model or theory is a representation of a "reality" which depends upon the purposes of the individuals and the tasks which they have (Wood-Harper, 1982). It is an assumption that the different "realities" can co-exist and it is the aim of knowledge elicitation to capture, in a single coherent framework, the perceptions of either one or several experts of a task domain.

The terms elicitation and acquisition tend to be used interchangeably. However, there is a clear distinction between elicitation and acquisition in the creation of a model. Elicitation is the process of developing a model by interviewing the experts and observing their environment; it is theory formation or model design. Acquisition is the process of collecting the detailed information (facts) which fit into the framework defined by the model. The difference between elicitation and acquisition is not always distinguished in some representation schemes such as semantic nets, production rules or other schemes which ignore the distinction between the model and the exemplars. This distinction is sometimes referred to as the intension (the model) and the extension (the exemplars or facts) of knowledge. For example, the modelling aspect of a data-base can be considered to be the rules that define the paradigmatic organisation (the schema) and the exemplar aspect to be the facts that fit into such an organisation (the data-base itself). Thus, how a man is defined and related to a mammal (say) is an intensional issue and is different from the extensional fact that Bill is a man. This distinction between intension and extension may seem arbitrary but as will be seen later it is a question of viewpoint.

Although these two views are independent and the acquisition of knowledge generally follows the elicitation stage, the two processes tend to overlap in practice. Briefly, this is because during elicitation the expert generally finds it easier to explain his knowledge in terms of specific cases and examples rather than in abstract terms. Sometimes the knowledge engineer needs to "induce" the model from a mass of detailed information and facts provided by the expert (Addis, 1978, Quinlan, 1979).

A model and its development play an important role in the interactions between experts and the knowledge engineer. The process of building a model gives a clear purpose to the discussion. The expert can visibly see a degree of achievement and progress as the model is developed. The model may, for example, represent entity sets and may place these entity sets into a context as the constraints are established between one entity set and another (the inter-entity set relationships) (Addis, 1985). Furthermore, as these entity sets are discussed and their attribute types determined, ambiguities in meaning and understanding

may be uncovered. Through the interpretation, updating and the running of the model, the knowledge engineer's understanding is continually being checked.

The questions arise as to what kind of knowledge is being elicited, how is this knowledge related to the model and what are the necessary forms of such a model. The nature of knowledge has been a subject of investigation for over 300 years. Descartes (1596-1650) was one of the first to apply formal arguments to the discovery of certainties in the field of human enquiry (Descartes, 1968). The purpose of his investigations and others' since was to provide an understanding of the distinction between knowledge and illusion. It has became clear that there is no simple answer to this distinction since our observations of the world are through senses that are highly selective. Arguments have been created that are supported by empirical evidence which show that this selection process is controlled by a range of predefined conditions. These conditions involve both the sensitivity of the senses to stimulation and the predisposition of the observer. In particular, observations of the world are dependent upon expectations that are derived from hypotheses that are accepted as being "true". (Trusted, 1979, Harre, 1972)

It is one of the purposes of this paper to make clear the nature of human knowledge and machine knowledge by considering philosophical views in terms of Artificial Intelligence (and hence knowledge-based systems). This paper is not concerned with all the issues explored by philosophy. However, there is a range of insights that has emerged from philosophical studies that is of direct importance to Artificial Intelligence. (Peirce, 1934, Peirce, 1934, Peirce, 1958) In particular, the classes of knowledge that have been recognised by philosophers can be equated with the structural forms of knowledge that are required for Artificial Intelligence programs. With this insight, knowledge for Artificial Intelligence programs can be seen as falling into three groups, "asserted knowledge", "inference" and "heuristic" (Figure 1).

## 2. ASSERTED KNOWLEDGE

### 2.1. Introduction

Asserted knowledge represents a posteriori knowledge of the world. It is knowledge that has emerged from both experience and the application of various forms of reasoning. Asserted knowledge can be subdivided into facts and hypotheses (Figure 1).

One of the ways of representing asserted knowledge in an Artificial Intelligence program is as a semantic net. Semantic nets are illustrated as labelled arcs and nodes. Each node represents a fact and each arc a relationship. The most common relationship is the ISA arc. The ISA arc has at least two distinct meanings. The first is an instance of some category or class of objects. Thus "Fred ISA man". The second is a subset of some category or class of objects. Thus "Bird ISA mammal". The first meaning is an

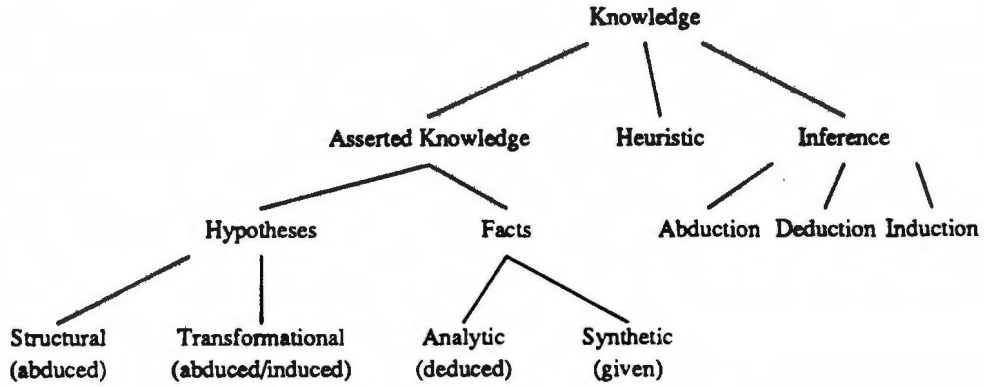



*Figure 1 Types of knowledge.*

example of a fact and the second meaning is an example of a defining hypothesis. There is no standard set of primitives for creating semantic nets and each task domain will define what constitutes an elementary arc and node. The approach in most cases is to make arcs a closed set of organisational (structural) relationships and nodes an open set of substantial (factual) relationships.

### 2.2. Facts

Facts are propositions that make statements about the world and can be either analytic or synthetic. Synthetic facts are propositions about the properties of objects (or events) that have been observed or given as true. Such propositions may be specific to a particular object or event or generalisations about sets of objects or events. Analytic facts are propositions that have been derived from facts (of either kind) through a process of deduction. Given a set of facts, it is possible through the application of hypotheses (rules for change and classification) and the rules of deduction to derive analytic facts; facts can thus be inferred from given or previously derived facts using a hypothesis and deduction.

### 2.3. Hypotheses

A hypothesis is a proposition that leads to the prediction of facts under specified circumstances. It must be likely in itself (it must be a "reasonable" proposition) and it must render facts likely (it increases the predictability of observing the existence of an object or event). Hypotheses can be simple and specific (eg. This 1 metre bar of iron will expand 1mm when heated from room temperature to 100 degrees C.), simple and general (eg. All people who drive red cars are accident prone.) or complex (eg. The Theory of Relativity.). Their purpose is to allow decisions to be made and future actions to be planned. Note that all the example statements are generalizations and assume a principle of consistency.

Hypotheses are propositions that describe the behaviour of the world and link facts together into some kind of structure that forms generalisations. Hypotheses fall into two categories, one describing a change of state (a transformation) and the other a class of objects or states (the paradigmatic organisation or structure). Transformational hypotheses are suggestive of a rule through which consequences may be deduced. For example, the hypothesis that "Objects lighter than air will rise if unattached" describes an expectation given specific and identifiable circumstances in the world such as having a lighter than air object. Structural hypotheses are definitions that describe the texture of the world; they are abstractions of the stimuli that impinge on the senses giving both generalisations and distinctions in order that objects and events may be classified.

## 3. INFERENCE

### 3.1. Introduction

Inference is a priori knowledge since without the ability to infer there can be no interpretation of the facts through hypotheses and there can be no new knowledge. The process of human enquiry requires at least three classes of inference. These are "deduction", "abduction" and "induction". Abduction is the act of insight that creates a hypothesis, deduction is analytic reasoning that gives a hypothesis life and induction is the process that gives support to a potential hypothesis.

### 3.2. Deduction

Deduction is well understood and the mathematics of valid reasoning methods forms the principles upon which machines can manipulate knowledge. The process of deduction cannot generate new knowledge since each step in the process is a tautology; it is a truth preserving mechanism. The conclusions must follow from the premises and are called analytic propositions. Note that "knowledge" which is deduced from a hypothesis and a given fact is an "analytic fact" (see Figure 1). These analytic facts are not generally obvious and the process of deduction reveals new views of the knowledge expressed by a system of hypotheses and facts. Such views can be used to make predictions or make statements about the unobserved world.

Deduction describes an acceptable way of reasoning that may be performed under heuristic control (see below). Deduction, as used for machine reasoning, may be a valid argument framework or an acceptable mathematical technique, such as the set of relational operations. Thus any formal procedure that is valid and tautological (truth preserving) will be deductive.

Tautologies can be represented as any valid well formed formulae (wff). The propositions P and Q are tautological if and only if P can replace Q without any change in the final truth conditions for all interpretations. An interpretation of a wff is an assignment of truth values to the component atomic propositions. A wff is valid if it is True under all its possible interpretations (otherwise it is invalid). It is in this way that a tautology preserves the truth.

An argument is a set of wffs (premises) followed by a single wff called a conclusion. An argument is valid only if the conjunction of premises (eg $P \wedge R$) implies the conclusion and this compound proposition is also valid (ie. $(P \wedge R \rightarrow Q)$ is valid). The validity of an argument is determined only through inspection of all the interpretations. Once an argument form (pattern of premises and conclusions) has been validated, then this form is independent of content (meaning). If the form of the premises can be matched to a subset of propositions then the conclusion can be added without loss of consistency. The conclusion is said to follow logically from the premises. An argument remains valid no matter what additional premises are supplied (monotonicity). Examples of valid argument forms are Modus Ponens (P, $P \rightarrow Q$ therefore Q), Modus Tollens ($\neg Q$, $P \rightarrow Q$ therefore $\neg P$) and, for machine reasoning, Resolution (a generalisation of Modus Ponens). (Chang, 1973)

Conclusions drawn from a deductive system are "closed" in the sense that no new knowledge of the world can result. The knowledge may seem new in that an individual may be startled by the results of applying the methods, but this only reflects the limitations of the human ability to naturally perform deduction. The results cannot be new knowledge since the links between the initial facts and the results are tautologies. What is new, of course, is the perspective; and the consequence of applying valid arguments is to observe the same knowledge from different points of view.

### 3.3. Abduction

Abduction is the process of creating a new hypothesis (new knowledge) when presented with puzzling phenomena. The combination of abduction and induction is often referred to simply as "induction"; it generates new knowledge of the world and for this reason is called ampliative reasoning. There are two types of abduction. The first type is concerned with the creation of new distinctions or generalisations; these distinctions or generalisations are the primitives (called percepts) from which hypotheses can be constructed. It is usually possible to find new distinctions since we are not limited to our senses alone. Instruments can be created (telescopes etc) that will interact with the world to any level of detail required. The structural component of knowledge (taxonomic knowledge) reflects the distinctions created through the first type of abduction (type 1). This knowledge references the features, attributes and categories that make up a given set of perceptions. This set of perceptions is such that it allows the formation of viable hypotheses that serve a particular purpose.

The second type of abduction (type 2) is hypothesis formation. This is the process of creating new transformational knowledge that can through deduction make predictions. How new perceptions are created, how hypotheses are formed and how the two types of abduction interact have not yet been fully explained.

The objective of hypothesis formation is to make the world appear stable and controllable through deducible predictions. At one extreme it is possible to make the world completely predictable and stable by making no distinctions (a trivial generalization). This procedure would not be very effective for normal survival nor in achieving some other predefined purpose. There are constraints imposed by the world that demand recognition if specific objectives are to be met. It is thus important to identify the significant distinctions in the world in order to meet these desired ends. It is equally important to avoid distinctions (specific generalizations) if a useful hypothesis is to be abduced. It is purpose that drives abduction and it is knowledge of the world that enables abduction.

### 3.4. Induction

Induction is the process of assessing the truth, the utility or the viability of a hypothesis. There are three classes (genus) of assessment or testing of a hypothesis (Figure 2). One of these classes of assessment depends upon the principles of random sampling and probability. There are three sub-classes of probability assessment depending upon the nature of the object sets. These sub-classes are: finite, infinite with known order, and infinite with quasi-random structure.

A random sample is drawn from the whole class and by a method, which if applied repeatedly, will in the long run give the result that collections of equal size and of a given distribution of properties will be chosen with equal probability. Probability is a ratio that involves the total number of objects and the proportion of the predesignated properties in the collection. Note that the properties (distinctions) are predefined. It is only with a finite collection that a random sample can be drawn. An infinite set may never be defined in the sense of "known". Only an infinite set of artificial objects (such as numbers) may be understood in finite terms.

Objects that are distinct and countable may belong to an infinite set. It is not possible to "know" such a set of objects. However, it is possible to know a finite subset and to suppose that such a subset represents the character of the whole set by the assumption of some order and continuity of the set. For example, the throwing of a die represents an infinite but countable set of events. We can assume that the behaviour of the die will always remain consistent and that any large sample will reflect the total character of the infinite set of events. In the case of the die we "know" from the nature of the object (the die) that its behaviour will be consistent for our purposes.

It is not always possible to "know" from the nature of the objects that their behaviour will be consistent. However, it might be assumed that the distribution of properties is uniform for samples of the infinite set. For example, the value of $\pi$ may be considered to be a quasi-random string of numbers. Samples may be drawn from the string to determine the the frequency of each number. It might be assumed that this distribution will hold for the calculation of $\pi$ to any level of precision.

The second class of inductive test has nothing to do with probabilities or improbabilities. There are observations which may support a hypothesis that describes the continuity of a process. For example, the tides in a certain bay are observed to rise every half a day and the hypothesis suggests that this will always be the case. It is a question of probability if these events are considered as infinite but countable. However, if the question is not about the likelihood of the tide rising every half day but about whether it always does occur every half day then sampling is not possible. Probability cannot justify a statement concerning the continuity of events since it requires only one failure to eliminate a hypothesis of this kind.

The third class of inductive test is where the sample is drawn from an aggregate which cannot be considered as a collection. It is a sample that does not consist of units that can be either counted or measured.

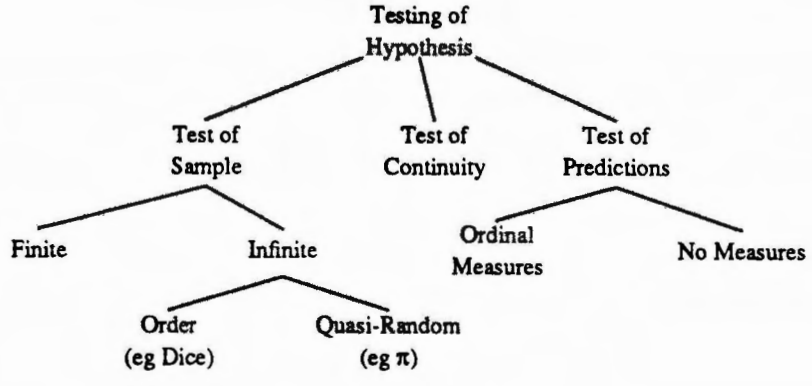



*Figure 2 Types of Induction.*

However, it is possible to order the samples in terms of greater or lesser quality. Under these conditions the size of the sample will not be important since no correction of error is possible. This class of test can be subdivided into ordinal measures and the assessment of a hypothesis through its predictions (no measures). The latter is the weakest form of induction. In the case of the kinetic theory of gases each improvement of the theory provides better predictions and a theory of greater utility. Such judgements cannot be sensibly quantified.

Induction in the restricted sense used in this paper depends upon the pre-existence of a hypothesis. The purpose of induction is to provide support for the hypothesis and it is the act of induction to specify an adequate test. Induction is not the test itself but the acknowledgement of what the test shows. This support is usually achieved by comparing the deduced predictions with the observed facts. The support may depend upon a statistical judgement or some method of quality assessment. It is important to note that it is the choice of method and the understanding of the assessment that is induction. In this sense induction is also dependent upon a priori knowledge of the world.

## 4. HEURISTICS

Knowledge of when to apply hypotheses to facts is referred to as heuristics. Heuristics are a means of controlling the application of deduction and are rules that order the choice of asserted knowledge for deduction (Pearl, 1984).

The heuristic control of deduction is dependent upon the problem to be solved. The discovery of heuristics can also be described in terms of a deductive process which references the knowledge-base as a set of asserted facts. Heuristics would seem to emerge from a different task domain than that from which the problem has been drawn.

## 5. KNOWLEDGE-BASED SYSTEMS

Competence consists of both knowledge about the task environment (modelled by structural hypotheses and facts) and the skill to perform the task (modelled by transformational hypotheses and heuristics). A knowledge-based or expert system is dependent upon a knowledge-base and an interpreter (see Figure 3).

The interpreter defines how deduction should be used to achieve the desired ends. In this limited respect, the interpreter is concerned with control by setting the goals of the system. The choice for control (ie the heuristics) is supplied by the knowledge-base.

A data-base system is restricted to making explicit only the facts and the structural hypotheses. The reason for this is that the storage of data within mass memory (eg Disc and Tape) is normally described in terms that are homomorphic to the representation of facts and the structural relationship of facts. The knowledge representation symbols for data-base design restrict the possible models to those that can be constructed within a data-base storage framework. Thus, the facts may be represented as records or tuples in a relation and the structure may be represented as record structures or relationships. The heuristic and transformational knowledge will be implicit in the application programs that use the data-base simply because this class of knowledge cannot be represented explicitly and directly in data-base terms.

## 6. REASONING ABOUT KNOWLEDGE

### 6.1. Introduction

A hypothesis is a proposition that leads to the prediction of facts under given circumstances. It has been shown that, given a knowledge-base, the only way in which that knowledge-base can change is either through the introduction of new knowledge from an external source or from the construction of analytic facts. This is because deduction is the only inference mechanism available for machines.

This restriction can apparently be overcome by considering the knowledge-base of the task domain as facts within an outer-knowledge-base. The outer-knowledge-base is knowledge about hypothesis reconstruction (see Figure 4). In this case there will be transformational hypotheses (outer-hypotheses) which are

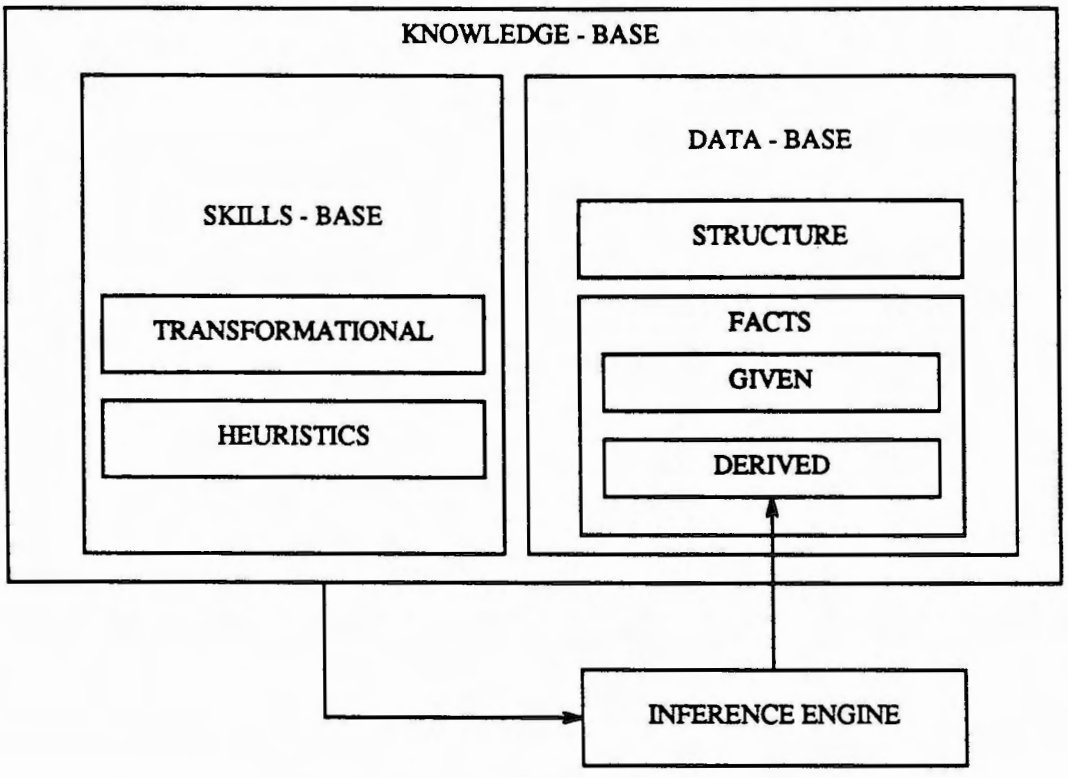



*Figure 3. The Structure of a Knowledge-Based System.*

concerned with how the knowledge-base of the task domain under consideration can form new hypotheses. These outer-hypotheses have been abduced to describe the learning behaviour that results in (hopefully) an improved knowledge-base for the task domain. The mechanism for learning is still deduction but because new hypotheses are being created the process of learning will appear to be abduction from the viewpoint of the task domain. Figure 4 illustrates two stages in a recursive knowledge-based system. It is recursive because the process of learning can occur at all levels.

### 6.2. Generalizing from Observations

A hypothesis is a proposition that renders facts likely by the application of deduction. A better hypothesis will be a simpler proposition and it will render more facts likely with greater precision. For a closed environment which is both countable and finite a hypothesis provides a reduction of entropy with respect to the analytic facts. A reduction of entropy means that the environment becomes more predictable where the prediction is concerned with a predesignated set of properties.

A hypothesis depends upon the percepts of the task domain. A percept is a given distinction in the world and a distinction depends upon the creation of a test that has a predefined domain of results. The distinctions have been abduced (type 1) from the task domain and they express a set of distinctions and generalisations as tests for a knowledge-based system. These tests act as filters on the senses (sensors, inputs) of the hypothesis user and as such represent a primitive set of concepts. Examples of percepts in knowledge-based systems are:

(1) Phillips (Phillips, 1978) discourse analysis where the sentence

"John chased Mary"

can be represented by

((TEST=Agent DOMAIN=Noun RESULT=John)

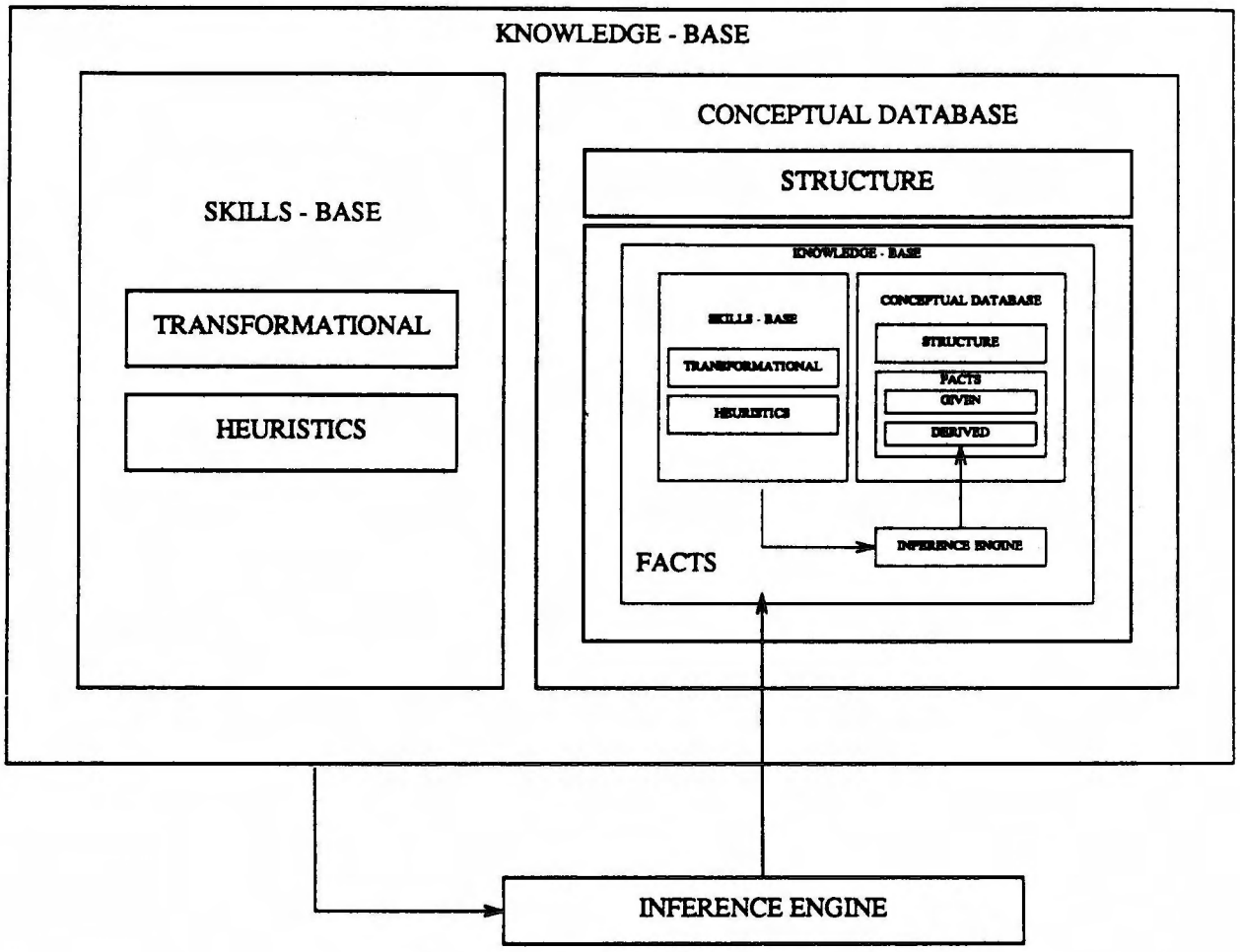



*Figure 4 A Recursive Knowledge-Based System.*

(TEST=Action DOMAIN=Verb RESULT=chase)
(TEST=Experiencer DOMAIN=Noun RESULT=Mary)).

(2) Hendrix (Hendrix, 1975) partitioned nets will describe

"Bolts attach a pump to a platform"

by

((TEST=Fastners DOMAIN={Bolts, Screws} RESULT=Bolts)
(TEST=Minor-part DOMAIN={Pump, Pipe, Cable} RESULT=Pump)
(TEST=Major-part DOMAIN={Housing, Platform} RESULT=Platform))

The percepts are laid down as fundamental in each of the above examples and have been chosen to express ideas that are appropriate to the purposes of the task domain. There is no absolute or universal set of primitives (percepts) although there may be a common set (such as ISA and its variants). The percepts chosen are abstractions that are understood to have meaning only to those who are familiar with the task domain. The percepts for a knowledge-base are part of the unproblematic background knowledge and it is through them that links to the world are made.

The percepts, represented as a TEST, DOMAIN and RESULT triple, are the elements from which both hypotheses and observations are expressed. If a specific observation of an object or event is considered to be a generalisation in that it is taken that similar circumstances will result in similar outcomes

then such an observation is itself a hypothesis (the consistency principle). It is a hypothesis because it takes on the role of a proposition that leads to the prediction of facts under given circumstances. The facts that are predicted are the analytic facts derived through the hypothesis and the observations (synthetic facts). There are thus predesignated classes of analytic and of synthetic facts that are defined by the hypotheses and the hypotheses are created in order to achieve a specific kind of prediction (eg. the actions of the expert).

Consider a finite collection of observations taken from some task domain. The purpose of these observations is to abduce a good hypothesis (abduction of type 2) that will predict the behaviour of an expert. These observations are orientated so that a situation is identified by a set of tests and each situation is followed by a consequence. To identify a situation it is also necessary to abduce the appropriate percepts (abduction of type 1). This step is usually achieved directly by simply asking the expert to identify the situation. For example, the expert is being observed performing his skill and we will record each important characteristic (as identified by the expert) of a situation and note his actions. It is the action of the expert we wish to predict (simulate) and to this end we need a hypothesis.

For example, suppose a situation is identified by a set of tests and results (eg symptoms in fault finding say). The tests are numbered and the results are either Y (yes) or N (no) and the actions of the expert are numbered as An. The set of observations (situations) is recorded and from the consistency principle they can then be considered in conjunction as a hypothesis thus:

Example 1.

$$
\begin{aligned}
&(1Y \wedge 2Y \wedge 4Y \wedge 7N \rightarrow A1)\\
&\wedge\ (1Y \wedge 2N \wedge 5Y \rightarrow A1)\\
\wedge\ &(1Y \wedge 2Y \wedge 4Y \wedge 5Y \wedge 6N \rightarrow A2)\\
&\wedge\ (1N \wedge 2Y \wedge 3Y \wedge 6N \rightarrow A2)\\
\wedge\ &(1Y \wedge 2N \wedge 5N \wedge 3N \wedge 7Y \rightarrow A3)\\
&\wedge\ (1N \wedge 3Y \wedge 6Y \rightarrow A4)\\
&\wedge\ (1N \wedge 2Y \wedge 3N \wedge 7N \rightarrow A4)
\end{aligned}
$$

Such a hypothesis can be used to simulate the expert by the performance of the tests and the application of deduction. A hypothesis, in this sense, is a model of the expert's behaviour.

This hypothesis is very awkward and it is only useful if the situation when it reoccurs is exactly as previously observed. However, it might be possible to simplify the hypothesis by reducing the number of tests to be done in order to identify any particular situation. To this end the hypothesis needs to be restructured according to some assessment principle that will ensure the logical coherence of the initial hypothesis and will also minimise the number of tests to identify a situation. Drawing upon established data sorting techniques, a method is to sift through the situations associated with each action and generate an optimum test sequence in the form of a multi-branching tree. The first node of the tree will represent the best test, where the best test divides the set of actions into as near to equal-sized and stable subsets of actions as is possible. The second set of tests in the next level down the tree repeats this for each subset, and so on until the final test in a branch isolates an action. The measure of quality for a test requires that the test is reliable, provides an equal division between the actions and concentrates on the most probable actions.

Techniques of pattern recognition show that there is a single complex measure which will balance all these requirements so that the 'best' test may be chosen. This single measure is the 'reduction in entropy per unit time' brought about by applying a particular test. (Addis, 1978, Addis, 1985) This particular technique is equivalent to the "rule induction" methods used for Expert Systems that have evolved from Quinlan (ie. the ID3 series) (Quinlan, 1979).

The result of applying this technique to example 1 is a hypothesis which now appears as:

Example 2.

$$
\begin{aligned}
&(1Y \wedge [(2Y \wedge 4Y) \vee (2N \wedge 5Y)] \rightarrow A1)\\
\wedge\ &((1Y \wedge 2Y \wedge 4N) \vee (1N \wedge 3Y \wedge 6N) \rightarrow A2)\\
\wedge\ &((1Y \wedge 2N \wedge 5N) \vee (1N \wedge 3N\ \ 7Y) \rightarrow A3)
\end{aligned}
$$

$$\wedge\ (1N \wedge [(3Y \wedge 6Y) \vee (3N \wedge 7N)] \rightarrow A4)$$

Example 2 is a better hypothesis than example 1 because it is simpler in some respects. It is not more predictive in that the same actions can only be deduced under the same situations as were originally observed. However, the technique can eliminate tests that are redundant to the identification of a situation and in that sense the hypothesis will be more general. Such generality will hence be more predictive since the recognition of a situation is not dependent upon an irrelevant test.

Consider what happens in the reconstruction of a hypothesis.

(1) Situations were recognised by asking the expert (whose skill we are attempting to simulate) what are the essential percepts. This avoids the problem of type 1 abduction. However, it does open up the problem that the expert can in fact identify a situation through a finite set of observations. This may not always be the case (Wittgenstein, 1953).

(2) A hypothesis of the expert's actions is constructed on the assumption that the expert's behaviour is consistent with identifiable situations. This assumption arises from an abduction of type 2 and the result (the initial hypothesis) is the application of deduction based upon this assumption.

(3) Each percept (a test) is examined and assessed according to its ability to identify (predict) a situation. It is induction that presupposes that an assessment drawn from the class "Test of Sample" is an appropriate assessment for this activity. It is induction that presumes that entropy is the appropriate hypothesis for this purpose.

(4) Using the results of the assessment then a new hypothesis is reconstructed from the old. The method for this reconstruction assumes that the tests can be done in any order and they are independent. The method used is the outcome of an abduction of type 2. The reconstruction is the result of applying deduction to the hypothesis embodied in the method.

In every case the act of abduction and induction has occurred long before the *rule induction* process is used. This is because there exists an understanding about the world which currently is found only in people. It is people who decide that such a method is appropriate and it is people who provide the abductive and inductive inference.

## 7. THE METHODOLOGICAL FALSIFICATION PROGRAMME

The process of developing a model or hypothesis can be considered as being equivalent to the development of any scientific theory. A final model should provide exactly for the data needed to be established within it; that is, it will cater for all the observations of the expert's behaviour. A scientific theory similarly tries to develop a model that will explain all plausible observations. The methodology of the Scientific Research Programme advocated by I. Lakatos (Lakatos, 1978) suggests that scientific achievement cannot be quantified by an isolated hypothesis or theory but by the shift in theory over time; that is, the Research Programme.

The Lakatos Programme proposes that an initial abduced hypothesis (H1) is supported by a set of observed corroborating facts (CF1). The hypothesis, H1, also predicts a set of novel, unobserved, theoretical "facts" EF1. This situation is depicted by figure 5. The facts (CF1) are the initial data given by the experts, some of which may be in a generalised form. For example, given the fact 'Sam is a cat' (CF1) and the hypotheses 'All mammals are warm blooded' (Transformational) and 'Cats are mammals'(Structural), then this will predict the fact that 'Sam is warm blooded' (EF1).

The unproblematic background knowledge consists of indisputable facts or axioms for which a hypothesis is not required. This would define the limits of the model and the fundamental percepts from which the model is constructed. Thus, in the domain of 'pump construction' (Hendrix, 1975) the primitive relationship 'fastner' need have no further description; it is a percept. Safety controls and conventionalism are the agreed methods for both the recognition of an observation as being true and for reasoning about these facts once established (eg. the perceptions of the user of the knowledge-based system and the deductive inference used by the system). Hypothesis H1 will remain stable until a second hypothesis H2 is proposed.

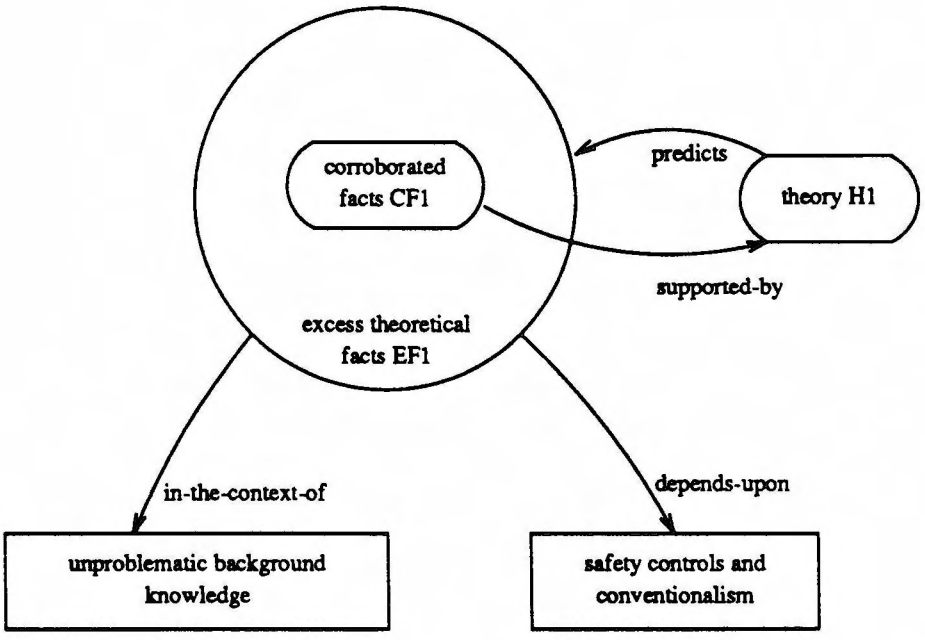



*Figure 5. The initial stage in the development of a theory*

Lakatos suggests that H2 can be of two types. Firstly, H2 can cause a progressive scientific shift. In this case the new hypothesis H2 must not only explain the previous success of the original hypothesis and corroborate some of the excess, uncorroborated facts of H1, but it must also predict novel facts; facts which either have never been thought of for inclusion in the set of unobserved, theoretical facts or have been contradicted by previous rival programmes. Thus, the fact 'Sam is a snail' would require an extension of H1 to H2. The progressive scientific programme is depicted by figure 6. Note that it is not sufficient simply to reproduce an expert's behaviour. There must also be the capacity to generalise.

In this way, a progressive programme for knowledge elicitation can be seen as a series of different hypotheses where each new hypothesis presents further insights into the given facts and the experts' performance. More generally, a progressive programme will consist of a series of branching sets of hypotheses (H2' H2") which are selectively trimmed according to certain criteria. A hypothesis (or hypotheses) is kept because it serves predefined purposes (such as design requirements) that go beyond mere accountability. In the case of knowledge elicitation one predefined purpose is the engineering of representation schemes within a specified knowledge-based system.

The choice of a symbol set by the knowledge engineer preselects the possible sets of hypotheses to be considered at each stage. Certain hypotheses can never be formed from these symbols because the primitives will not provide for the representation of these hypotheses. Other hypotheses will be preferred because the primitives form simple structures. The choice of primitives (symbol set) is crucial for efficient design. They should be selected so that simple representations of knowledge will produce simple implementations of a knowledge-based system.

The second type of programme is non-progressive. In this case NH2 is purely an extension of the original hypothesis through the addition of auxiliary propositions. These auxiliary propositions effectively confine the original hypothesis to take into account only the corroborated (or potentially corroboratable) facts. In this way the new set of unobserved, deduced facts (analytic facts) becomes a subset of the original. This may be depicted as in figure 7.

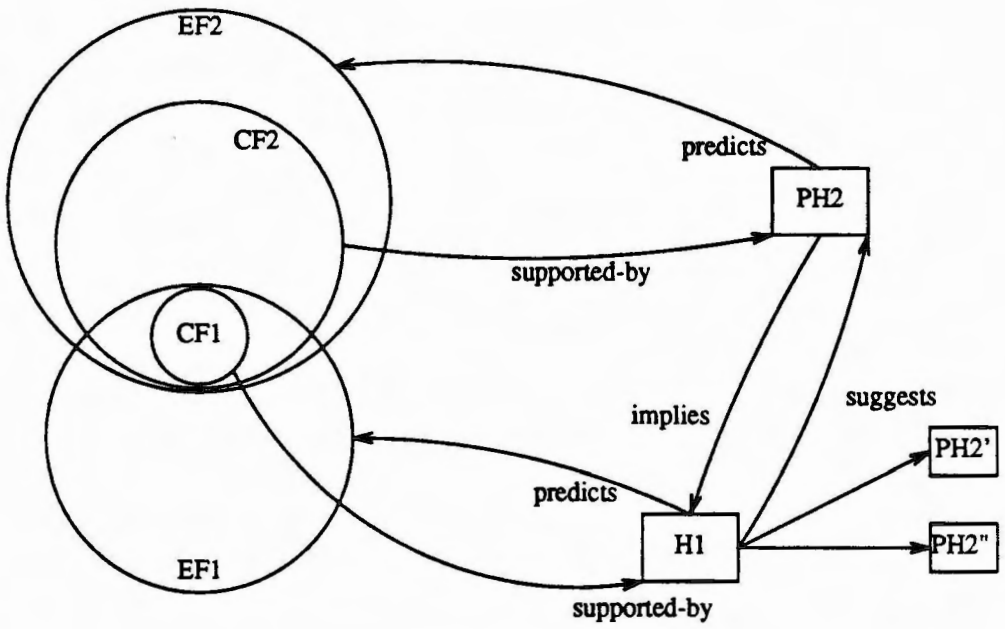



*Figure 6. A progressive scientific programme*

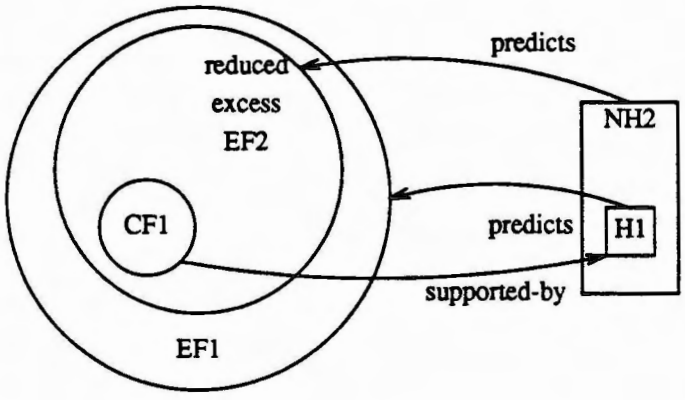



*Figure 7. A non-progressive scientific programme*

A hypothesis cannot be rejected by an isolated observation that conflicts with it simply because the hypothesis does not account for some observations. Instead, the hypothesis must be replaced by an improved hypothesis that accounts for more observations with greater commitment, in a manner that is progressive and where the hypothesis can imply new potential facts that, as of yet, are unobserved. There is the additional requirement that the implementation of the knowledge-base be simple.

Using this methodology, in a theory for elicitation, the corroborated facts are the cases or examples supplied by the experts. These corroborated facts are supported by the simplest model (the simplest hypothesis Hn) in agreement with the experts. The clarity and coherence of the structure of the model will

dominate the reasons for its acceptability since such a structure will normally result in good systems design. The model will also predict new, unobserved facts that can be used to question the degree of precision of the current state of understanding held by the knowledge engineer.

## 8. CONCLUSIONS

Different types of knowledge and an outline of hypothesis formation have been described. It has been argued that the symbols chosen for knowledge representation play an important role in the kind of knowledge elicited where it is presumed that this knowledge will result in a computer system and that the representation will be the first stage of system design. From the analysis in this paper there are several characteristics of knowledge elicitation that can be expected.

Knowledge elicitation is the creation of a model which displays an "ideal" expert's behaviour. The model is normally in the form of a compound proposition. The "ideal" expert is drawn from an amalgamation of one or more experts who perceive a coherent view of their task domain. Since the model is formal it can only be activated through deductive inference and it thus will be specifically orientated towards those tasks that require deduction. The domains for which such a model would be easiest to create are those that are clearly definable. Example domains are diagnostics and legislation.

A model that imitates the expert should also be capable of generalisation; it should be able to show new behaviour patterns. The kind of model that would satisfy the capacity for new behaviour is a model that captures the "grammar" of the expert's activities. Such a model is in the intensional form. Many of these models are based upon linguistic theories and are presented as a syntax of actions (Johnson, 1983, Angluin, 1982, Muggleton, 1987). Other models are defined in terms of constraints (Addis, 1985, Elleby, 1987).

Behaviour that involves learning will require a model which must include at least two levels of operation. The first level will be a model that defines the behaviour of an expert and his actions in a task domain. The second level will be a model that defines how the first model should adapt. The second model has a task domain concerned with learning in a specialised context. The first model, in part, is an 'extension' of the second. The change effected on the first by the second is a simulation of type 2 abduction. Pask (Pask, 1976), from the cybernetics' view of intelligent systems, has argued for the necessity of a two level system.

The models of behaviour will be limited by the percepts from which the propositions are created. These percepts must be understood by the user of the knowledge-based system. The meaning of each word used to represent a primitive within a knowledge representation scheme is a reference to a percept. This reference must remain stable during the life time of the knowledge-based system. Techniques for discovering these percepts through interviews have been developed and in particular personal construct theory is based upon determining the unique view that an expert has of his task domain (Gaines, 1980).

Systems that can react directly with the world through sensors which are selective can be assured of the meaning of their percepts. This is because the referent is used directly. Such systems should also enable the possibility of new percepts. New percepts are possible during direct interaction with the environment because "reality" manifests itself through constraints (ie. the results of testing the environment). Constraints and hence distinctions in the world can then be discovered through the generation of tests on the environment. These constraints can then be directly translated into actions that may or may not achieve the defined purpose of the system; the sensors are part of the feedback loop that involves the complete system. Constraints are sometimes referred to as *laws* or *causal propositions* (Trusted, 1979). The simulation of type 1 abduction requires at least the ability to be able to discover new relevant distinctions by interacting directly with the environment. Further, type 1 abduction requires the capacity to link these distinctions with those actions that will ensure the achievement of preset objectives.

The formation of a model will depend upon the creativity and skill of the knowledge engineer as well as the representation scheme he has selected. The process of creating the model will cycle through two modes. These are progressive and non-progressive model formation. The progressive mode will occur when a new set of percepts is acknowledged. This new set may retain some or all of the previous set of percepts. The new set will be the basis of a new model that provides an improved description of the

expert's behaviour in a simpler manner. The non-progressive mode is where no new percepts are acknowledged but new auxiliary propositions are added in order to cope with the range of observations. The knowledge engineer will then use the model to generate questions ( tests, experiments) that are presented to the expert. Conversations with the expert are also used to test the model by making predictions of his possible answers (teachback) (Pask, 1976). The model, although developed under these conditions, may stay in a non-progressive mode for sometime.

Auxiliary propositions are formed during these conversations in order to improve the model by a better definition of the expert's behaviour. However, this kind of improvement can create a more complex model that will result in design problems for the knowledge-based system. At some stage in the development of a model it can deteriorate by either losing generality or becoming incapable of expressing important elements of behaviour. If new percepts are not forthcoming then a compromise would have to be made between the experts, system users and system designers.

Knowledge elicitation is a complex issue that is balanced between a science of theory formation, the science of design and the science of human behaviour. This paper has touched upon the theoretical components of the problem by suggesting a framework through which the different approaches to knowledge elicitation may be compared.

## 9. ACKNOWLEDGEMENTS

I would like to thank my students for the many interesting conversations about knowledge and knowledge-based systems. In particular I would like to acknowledge the contributions made by Mark Nowell to the ideas and insights that have emerged in this paper. My guide through the works of Pierce has been Dr. J. Tiles. However, the interpretation and any errors are mine alone.

**References**


Addis, 1978. Addis, T. R., "A Feasibility Study of RAFFLES," ICL (RADC) Technical Note TN78/3, August (1978).

Addis, 1985. Addis, T. R., *Designing Knowledge-Based Systems*, Kogan Page (1985). ISBN0-85038-859-7

Angluin, 1982. Angluin, D., "Inference of Reversible Languages," *Journal of the Association for Computing Machinery* **29**, pp.741-765 (1982).

Begg, 1984. Begg, V., *Developing Expert CAD Systems*, Kogan Page (1984).

Chang, 1973. Chang, C. and Lee, R. C., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press (1973).

Descartes, 1968. Descartes, R., *Discourse on method and Meditations*, Penquin Books (1968). A Translation with an Introduction.

Elleby, 1987. Elleby, P. and Addis, T.R., *A Conceptual Model For Transaction Integrity*, MILCOMP 87, Military Computers, Graphics and Software (1987).

Gaines, 1980. Gaines, B.R. and Shaw, M.L.G., "New Directions in the Analysis and Interactive Elicitation of Personal Construct Systems," *International Journal of Man-Machine Studies* **13**, pp.81-116, Special issue on Advances in Personal Construct Technology (1980).

Harre, 1972. Harre, R., *The Philosophies of Science: An introductory survey*, Oxford University Press (1972). ISBN 0-19-888056-1

Hendrix, 1975. Hendrix, G.G., "Partitioned Networks for Mathematical Modelling of Natural Language Semantics," Technical Report NL-28 (1975). Department of Computer Science, University of Texas

Johnson, 1983. Johnson, N.E., *Elicitation and Representation of Children's Arithmetic Knowledge*, Chelsea College, London (1983). Thesis submitted for degree of PhD

Lakatos, 1978. Lakatos, I., "The Methodology of Scientific Research Programmes," *Philosophical Papers* **1** , Cambridge University Press (1978).

Muggleton, 1987. Muggleton, S. H., "Inductive Acquisition of Expert Knowledge," University of Edinburgh (1987). PhD Thesis

Pask, 1976. Pask, G., *Conversation Theory: Applications in Education and Epistemology*, 1976.

Pearl, 1984. Pearl, J., *HEURISTICS: Intelligent search strategies for computer problem solving*, Addison-Wesley (1984). ISBN 0-201-05594-5

Peirce, 1934. Peirce, C. S., "Scientific Method," in *Collected Papers of C. S. Peirce*, ed. A. W. Burks (1934).

Peirce, 1934. Peirce, C. S., "Pragmatism and Pragmaticism," in *Collected Papers of Charles Saunders Peirce*, ed. P. Weiss, Harvard University Press (1934).

Peirce, 1958. Peirce, C. S., *Charles S. Peirce: Selected Writings*, Dover Publications Inc (1958).

Phillips, 1978. Phillips, B., *A model for Knowledge and its Application to Discourse Analysis*, University of Illinois, Department of Information Engineering KSL-9, 1978.

Quinlan, 1979. Quinlan, J. R., "Discovering Rules by Induction from Large Collections of Examples," in *Expert Systems in the Micro Electronic Age*, ed. D. Michie, Edinburgh University Press (1979). ISBN 0-85224-381-2

Sowa, 1984. Sowa, J.F., *Conceptual Structures: Information processing in mind and machine*, Addison-Wesley (1984).

Trusted, 1979. Trusted, J., *The Logic of Scientific Inference: An Introduction.*, The Macmillan Press Ltd (1979).

Wittgenstein, 1953. Wittgenstein, L., *Philosophical Investigations*, Basil Blackwell (1953).

Wood-Harper, 1982. Wood-Harper, A.T. and Fitzgerald, C., "A taxonomy of current approaches to systems analysis," *Computer Journal* **24**(1) (1982).

# MEDIATING REPRESENTATIONS IN KNOWLEDGE ELICITATION

N.E. Johnson
Department of Computer Science
Brunel University
Uxbridge, UB8 3PH, U.K.

## 0. INTRODUCTION

A Mediating Representation for knowledge elicitation is a computer language independent notation used as a conceptual aid in synthesising knowledge from talk with experts. It 'mediates' between verbal data and standardised knowledge representation schemes found in AI software development environments. For the group at Brunel the major Mediating Representation is the Systemic Grammar Network (Halliday 1978, Winograd 1983 and Bliss, Monk & Ogborn 1983). This formalism is a key feature of the process of abstracting and conceptualizing expertise in a domain. It provides us with the means to prepare an explicit description of the knowledge in a definitive form for discussion within the KBS development team. It also allows us to attempt validation of would-be-knowledge as part of a more rigorous approach to knowledge elicitation (Johnson & Johnson 1987).

Much of this paper is devoted to outlining reasons why the notion of a Mediating Representation is useful (section 1) and what could be expected of an example of a Mediating Representation (section 2). Eventually, in section 3 two examples of Systemic Grammar Networks (SGN) are given. The first is a toy example (Figures 3 and 4) to demonstrate how a Mediating Representation may be used during knowledge elicitation. The second example (Figure 5) is a real SGN included to show that how even a fairly simple example can capture features of the domain not easily represented explicitly in the common knowledge representation languages of first generation expert systems. Glancing through section 3 before sections 1 and 2 is reccommended for ease of reading.

## 1. WHY BE INTERESTED IN MEDIATING REPRESENTATIONS?

Experience with knowledge elicitation in expert systems has led to two major reasons for being concerned with Mediating Representations, namely,

the nature of knowledge elicitation

to provide safer routes to the development of second generation expert systems.

### The Nature of Knowledge Elicitation

Elicitation involves creating an environment where an expert, and others, can generate some kind of description of their activities which the knowledge engineer comments upon, analyses and moulds into a body of 'knowledge'. Thus knowledge elicitation is not the discovery of heaps of mature, internalised cognitive structures; nor is it the mapping of

incompletely expressed concepts into the seductive precision of a formal system. It is closer to a learning or research activity where one, usually a knowledge engineer, comes to understand something of the concerns of the other (the expert). With varying degrees of appropriateness, the knowledge engineer actively creates the knowledge from a sea of qualitative data produced during their meetings. The two stages of elicitation, raw elicitation (interview, problem solving etc) and knowledge analysis (getting it down on paper) are distinct but do co-occur and knowledge engineers need skills in both. Skill in the use of Mediating Representations is an essential part of the knowledge analysis stage. For a description of common pencil and paper methods see Martin & McClure (1986) and SDL INSIGHT Study 6.

Second Generation Expert Systems

The experienced developer of first generation expert systems will have become adept at 'rule spotting' and feels no need of any extra representation device beyond conceptual hierarchies and decision trees. In this situation the rule base is likely to be the only substantial version of the knowledge to survive the project. So, only when the system is to be enhanced, ported to another machine or the original rule base writer moves on, will the KBS team wish they had a machine independent form of the knowledge to reimplement the system. Redoing the knowledge elicitation is a known, if desperate, solution to a lack of adequate knowledge documentation!

Even more difficult is the problem of the full custom knowledge engineer, working in a new domain and aware that a first generation architecture, typically an expert system shell, may not be adequate. What is needed is a facility for capturing what may be relevant and preserving it until a selection of implementable knowledge can be made. Using standard knowledge representation languages (Production Systems, Frame based languages, Associative Semantic Networks, Lisp like structures etc) is already to prejudge and thereby eliminate features of the expertise or knowledge which, with hindsight, ought to have been implemented. There is a clear parallel here with good software engineering practice in that care taken during specification (= elicitation) will pay dividends in time and effort during the later stages of prototype implementation.

To round off this section there are two remarks made to indicate the boundaries of the notion of a Mediating Representations. Firstly a Mediating Representation is not a 'paper knowledge base'. The paper knowledge base is a more restricted form of representation of some of the data from elicitation. It is a computer based representation scheme, perhaps with loose collections of 'other relevant information'. A paper knowledge base is a focus of activity during elicitation but the aim is to provide a communicable representation of the knowledge base only. A Mediating Representation is the major tool in constructing a machine-independent statement of the knowledge and associated background information called a Competence Model (Keravnou & Johnson 1986). The aim is to provide both a medium of communication between members of the team and a 'grammar' of the expert's task.

Secondly, a Mediating Representation is not a version of the linguists', and latterly knowledge engineers', notion of 'intermediate representation'. An intermediate representation is exactly what it says, a representation which only exists between flanking representations and is bound to them by clearly defined projection rules which map one representation to the next. It is obvious to anyone engaged in knowledge elicitation that the ability to translate or map something as

philosophically troublesome as 'knowledge' or 'expertise' from one representation form to another is not a task worth undertaking in this lifetime! So intermediate representation is indeed an unfortunate choice of words. We prefer the terms Mediating Representation to convey the sense of synthesis and coming to understand through the representation which is constitutive of knowledge analysis.

## 2. WHAT IS EXPECTED OF A MEDIATING REPRESENTATION?

### Explicitness and Expressiveness

Terms which express formal features of a representation scheme vary in how well they reflect or make transparent features of the data in question. A pre-selected collection of terms (the vocabulary of set theory, the family of terms relating to actors and agency or 'is-connected-to' type links) can be well defined but may not be appropriate to express the detail or colour of expertise. Special purpose terms made up of slots or well chosen names will be more expressive but of course can be simply idiosyncratic and therefore not explicit. For example, the basic If-Then format of rules in expert systems shells is often too bland to reflect explicitly all the shades of knowledge we may want to implement. (See compiled knowledge in MYCIN Rules e.g. Keravnou & Johnson 1985.) The richer source of terms in Associative Semantic Networks looks more promising but such networks can have as many types of nodes as types of links and so resemble conceptual spaghetti.

### Economy and Power

This is a notion borrowed from linguistics that says a grammar should be economical in the sense that it assigns structure to all and only the sentences of a language. Thus our mediating representation must capture all the relevant facts, opinions and relationships in the domain and exclude nonsense and contradiction. (Relevant here means with respect to the expert's conception). Representation schemes which generate conceptual absurdities or errors of fact are generally too powerful for the data. Thus an induction algorithm, which even on a restricted set of examples still generates unsympathetic rules, is not providing a good grammar (underlying structure) of the knowledge.

### Communicability

The major distinguishing feature of a Mediating Representation scheme and a Knowledge Representation language is the purpose to which each is put. A Knowledge Representation language is a machine oriented scheme where the demands of implementation are entirely appropriate. The purpose of a Mediating Representation is to provide a 'grammar' of the knowledge for communication between humans in order that appropriate choices of architecture, knowledge representation language and MMI features may be made. It is suggested here that a graphical format is necessary, although never sufficient. Hence it comes as no surprise to hear that experts find critiquing individual rules or other forms of pseudo code to be an uncongenial experience. Decision trees seem to fare better, but only when pruned! It is now even possible to display knowledge bases in the form of semantic networks on screen with integrity preserved under change. Unfortunately the facility to draw and redraw complex diagrams on screen does not guarantee that well structured accounts of knowledge will emerge. A picture of knowledge, no matter how detailed, cannot be a grammar. Nor

will it necessarily make knowledge transparent to anyone else on the KBS team. Thus indiscriminate use of sophisticated graphics editors will not make knowledge elicitation, as a conceptual task, very much easier.

Guidance in Knowledge Analysis

In the previous paragraph it was suggested that a sophisticated graphics editor, although desirable, can lead the unwary into producing visually and conceptually confused 'pictures' of knowledge. For the notion of Mediating Representation advanced in this paper we too require the 'pictures' to be visual displays. But they are to be displays of grammar like structures which reveal important features of the qualitative data arising from elicitation not just human oriented displays of machine readable code. For the sake of good practice in knowledge elicitation we require the facility to be able to test the structure against the data for fit and descriptive adequacy. The toy example of the next section shows how this may be done in practice. In principle, we require the Mediating Representation to do two things:

support the process of deriving data driven concepts

generate a description of the knowledge which can be tested against the original data for completeness and soundness (and subsequently against external sources).

In the first requirement, the Brunel group have adopted an approach based on Grounded Theory (Glaser & Strausss 1967) whereby transcripts of interviews are read closely with much indexing, collating and memo writing to generate simple abstract terms which lift the analysis one step from the transcripts but with the analysis still 'grounded' in the original data. This is one way of tackling the gritty problem known to social scientists as 'forming data driven concepts' or 'qualitative data analysis'. In these circles the problem is perceived to be one of analysis at a conceptual or semantic level. In knowledge engineering the problem has often been approached through Protocol Analysis ot 'transcript analysis' systems (e.g. KEATS) which only offer protection to syntactic analysis. These approaches would best be regarded as tools to clean up after a conceptual analysis. It can be argued that before entering any of the current KBS development environments there is always a conceptual analysis stage which is as yet unsupported and for which a Mediating Representation is essential. For it seems that most knowledge engineers make fundamental choices about the character of the expertise well before entering the KBS environment, usually driven not by the data, but by what can be implemented in that particular environment.

Hence, for example in KADS terms (Breuker & Wielinga 1984) we are persuaded to use a 'model' to analyse the knowledge where the notion of models and levels of knowledge rests firmly upon a computer architecture which has never been shown to resemble a model of expertise. Looking to cognitive psychology to help is tempting but as yet not very helpful. A mature approach to KBS development based on Personal Construct Theory (ETS or PLANET) uses cognitively real representations (in constructs) but falls far short of being able to represent a structure of an expert's tasks - a primary requirement in the development of second generation expert systems. See Clancey 1984, Chanderasekran 1986 and Keravnou & Johnson for more on task structures.

The solution in our projects is to use the approach of grounded theory to generate names of distinctions in the domain and expertise and to

encapsulate those in a Systemic Grammar Network (SGN). The SGN can be traversed to generate'descriptions' of events or circumstances which ought to represent genuine points of interest in the domain. For example, an SGN which characterises strategies of chip designers could generate descriptions of possible histories of design decisons. These possible histories are suggested by and checked against the original interview transcripts and subsequently shown to other chip designers to 'test' for completeness and soundness. These networks display, in a concise graphical form, nodes whose names reflect (expressively!) the salient features of the domain and expertise. The few and simple links have the advantage of making the displays easy to absorb. The pseudocode generated by traversing the SGN or picking out combinations of terminal items should describe all and only possible events or 'objects' in the experts task. Thus our SGN's with supporting glossaries are custom built to provide a grammar of the task. By accepting the discipline of producing a grammar it is possible to be guided in a principled fashion through the knowledge analysis phase. By repeatedly asking the questions

> Does our expert approve of the choice of terms used in the SGN?
>
> Does the total set of combinations (the paradigms) adequately represent the possibilities the expert described during elicitation?
>
> Do the sequences of pseudocode generated by passing through the network generate sensible characterisations of real events or objects with respect to the domain?
>
> Do the 'failures' say something about the SGN or about holes in the elicitation data?

a series of SGNs will evolve which capture much more of the flavour of the expertise and domain than is normally believed necessary. The short argument is that if explicitness is desirable, then a Mediating Representation is an essential tool and it should both guide the process and provide a fertile structural format.

## 3. SGNs - AN EXAMPLE OF A MEDIATING REPRESENTATION

A Systemic Grammar Network is a well defined system, based on the idea of choice, with no standard rules of application. Essentially the notation provides an empty structure into which a chosen domain of knowledge may be fitted. The 'content free' of networks is their most useful feature in knowledge analysis. Like other networks, a systemic network is a graph composed of nodes joined by links. All nodes are names invented by the analyst but suggested by the data. The most commonly used links are of two types, a mutually exclusive choice, denoted by a straight bracket and a co-occuring choice, denoted by a curly bracket as in Figure 2. (There is also a recursion symbol and a conditional link.)

For the purpose of illustrating the heuristic use of SGNs imagine a simple classification problem of characterising the range of periodicals and newspapers found on newstands outside a London tube station. Our initial data is a collection of publications from one newstand. The aim is to draw an SGN which will represent the distinguishing features (and attributes) of all the publications in such a way that we may individually

characterise each example. By looking at a random collection (say The Times, The Sun, Country Life and Time Out) we could form data driven distinctions such as




Figure 1

and name the distinctions STATUS and FORMAT concatenating the distinctions thus

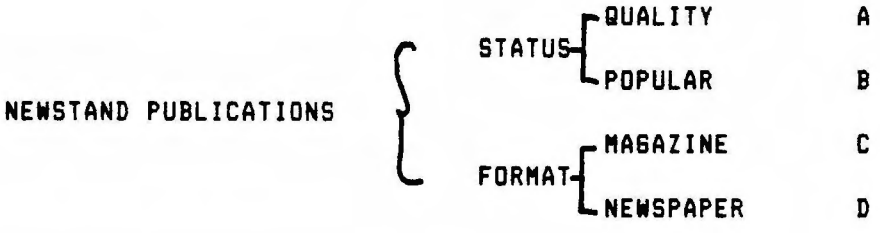



Figure 2

which can be read from left to right as

> NEWSTAND PUBLICATIONS have two (co-occuring) features, a STATUS which is either QUALITY or POPULAR and a FORMAT which is either MAGAZINE or NEWSPAPER.

Notice the embryonic network was not constructed from left to right. In fact in this trivial case names for the extreme nodes were first chosen and then a name for the middle distinction. This is common practice and at the heart of the heuristic role of SGN writing in knowledge analysis. For while names for the extreme terms may be obvious, the hardest and so most profitable mediation is done through sorting out the middle terms. The 'paradigm' of this network is the set of combinations

| | |
|---|---|
| A(and)C | quality magazine |
| AD | quality newspaper |
| BC | popular magazine |
| BD | popular newspaper |

which in the sense of section 2 must 'mean' something in terms of the data. In this paradigm we have a potential characterisation of any individual publication. The most important question is 'Does the paradigm represent all and only the possibilities?' and for that we need to check with the newstand.
AC is realised as Country Life; AD as The Times; BC as Woman's Own and BD

as The Sun. So far so good! But what about BYTE, Buttons or the Jewish Chronicle? Other data driven distinctions are now suggested. Special Interest Group (SIG) is one example. This may be appended as in Figure 3 where SIG is a distinction of the same rank as STATUS and FORMAT or as in Figure 4 where SIG is a distinction closer to STATUS.

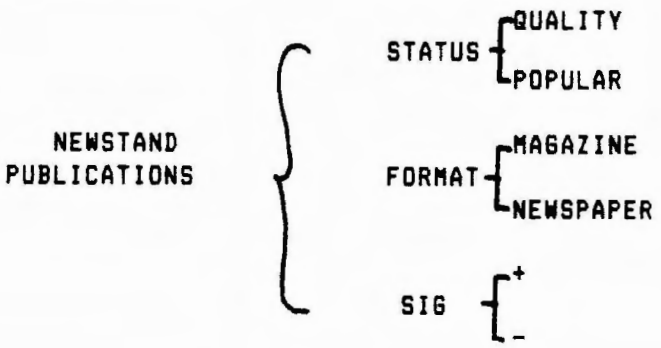



Figure 3

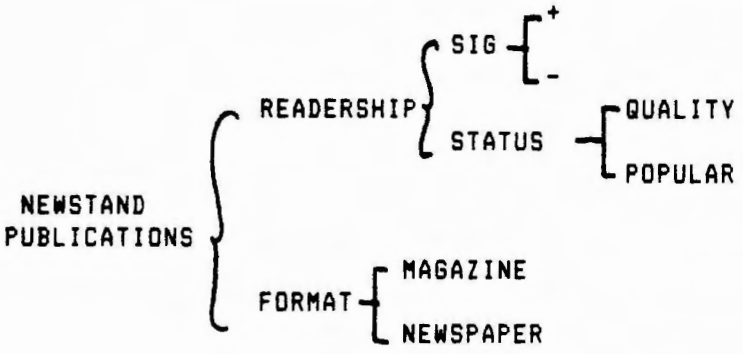



Figure 4

A choice is made between Figures 3 or 4 on the basis of which structure best reflects the natural divisions of the domain as well as which structure is most economical. At this point the analyst would need to ask for the natural divisions of buying periodicals (seek more data). Thus BYTE is a quality, SIG magazine; The Sun is a popular, not SIG newspaper and Buttons still does not fit! Clearly we need to add more distinctions, COMIC or SHELF LIFE (daily, weekly monthly, occasional) and eventually by continually testing paths and generating paradigms we could arrive at a one page SGN which was adequate for characterising the limited domain.

Without checking out paradigms with the data the focus or purpose of the network will shift and the structure degenerate into a standard AND/OR graph or decision tree. Any given publication may be characterised by a path through the network but it will be hard to see the range of

possibilities (the paradigms) in that form of degenerate network.

In Figures 3 and 4 notice that the names of the publications do not appear in the network. The network characterises the domain, it does not generate a representation of the domain. Of course this could be done in another SGN. Also other SGN's could be built to represent buying strategies or the vendor's layout policies. All such SGN's would have one thing in common, that the content is in the choice of nodes and not in the pre chosen links. For communicability a glossary of the node names must be provided. Fortuitously this is a welcome discipline in its own right. For a more complete account of the application of SGN's see Bliss, Monk & Ogborn (1983).

Figure 5 is an example of an SGN produced during knowledge elicitation with an architect and an occupational therapist involved in designing for the disabled (Funes 1987). This SGN is one of a series produced whilst trying to capture the knowledge required to assess the needs of disabled clients and reccommend changes of abode, adaptations to the existing home or specific aids.

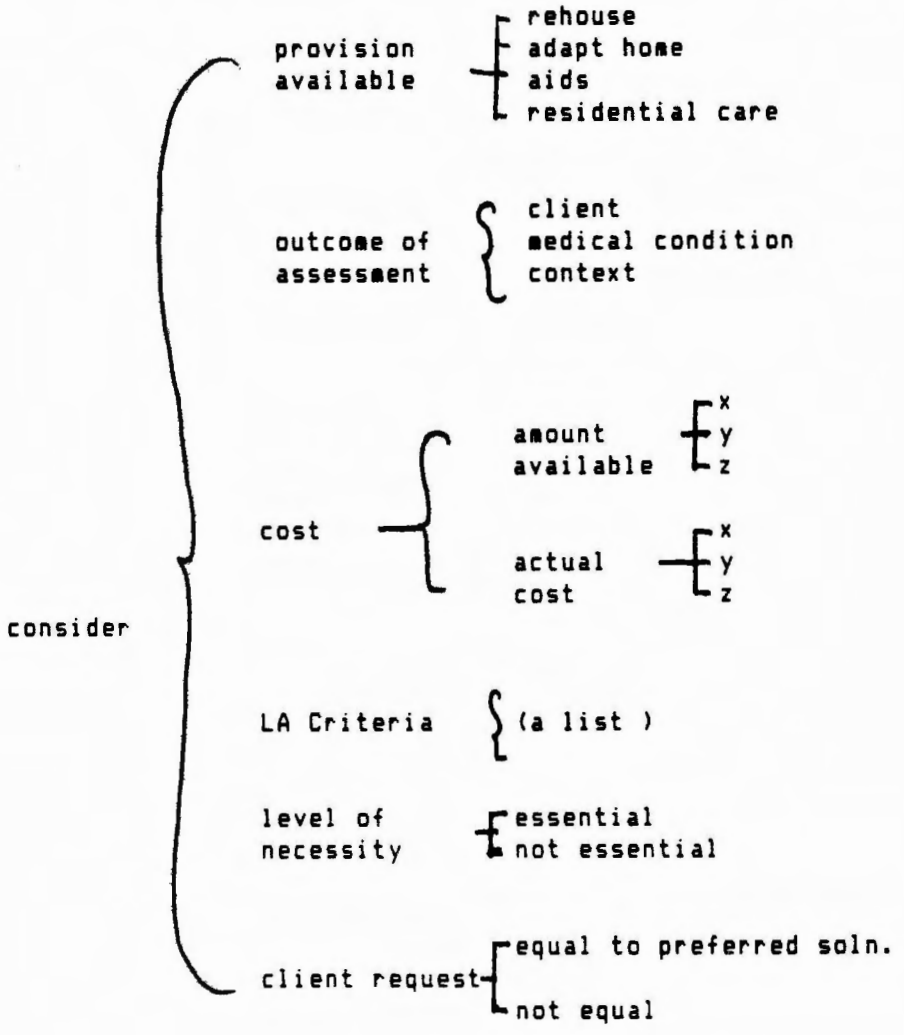



Figure 5

The purpose of Figure 5 was to represent all the considerations which must be taken into account from the point of view of the experts (architect and OT). In the time available it was not possible to do more than state each of the six left hand distinctions as being of equal importance. There is a 'hole' in the network due to a hole in the data. More elicitation could tease out a more differentiated structure. The fine detail of cost or medical condition is deliberately not represented at this level. That might be a hole in the network but not due to an absence of data. The analyst would argue that she chose the appropriate level of detail to make a point about the generic structure of the problem.

Looking back to the concerns of section 2, while Figures 3 and 4 are not bona fide SGN's, Figure 5 is subject to the scrutiny. From the experts point of view the terms of the network are grounded in the data and are expressive enough to reflect salient features. Even a cursory glance at the paradigms of the network shows that they make reasonable sense. However it is quite possible that elegance of structure has been sacrificed in favour of ease of reading. As a result of this it is likely that real flesh and blood client's histories do not exist for every single paradigm. Said more abstractly, this SGN may not be as economical as possible and is probably too powerful for the data. But it does (and did) function as a medium of communication. After all it is less arduous to 'interrogate' the SGN than to interrogate the original transcripts. Finally, the very existence of this and its sister SGNs is at least suggestive evidence of the ability of SGNs to guide knowledge analysis.

## BIBLIOGRAPHY

Bliss, J. Monk,M. & Ogborn, J. (1983) Qualitative Data Analysis: a guide to the use of systemic networks, Croom Helm, London.

Breuker, J. & Wielinga, B. (1984) Interpretation Models for Knowledge Acquisition. In O'Shea, T. (ed) Advances in Artificial Intelligence, North Holland.

Chandrasekaran, B. (1986) Generic Tasks in Knowledge Based Reasoning. IEEE Expert, 1,3 pp23-32.

Clancey, W.J. (1984) Acquiring, Representing & Evaluating a Competence Model of Diagnostic Strategy. HPP Memo 84.2, Stanford University. To appear in Contributions to the Nature of Expertise, Chi, M. Glaser, R. and Farr, M. (eds).

Funes, M. (1987) Knowledge Elicitation using teachback Methodology and SGNs, MSc Thesis (Intelligent Systems) Brunel University, Uxbridge, UB8 2PS UK.

Glaser, B.G. & Strauss, A.L. (1967) The Discovery of Grounded Theory: Strategies for Qualitative Research. Chicago, Aldine.

Halliday, M.A.K. (1978) **Language as a Social Semiotic**, Edward Arnold, London.

Johnson, L. & Johnson N.E. (1987) A Knowledge Elicitation Method for Expert Systems Design. **Int J. Systems Research and Info Science. Vol. 2.** Gordon & Breach, London.

Johnson, N.E. (1985) Varieties of Representation in Eliciting and Representing Knowledge for IKBS. **Int. J. Systems Research and Info. Science** 1,2 pp69-90.

Johnson, N.E. (1987) Knowledge Elicitation In Practice, **INSIGHT STUDY NO.6,** Systems Designers Scientific, Pembroke Broadway, Camberly, Surrey GU15 3XD, U.K.

Keravnou, E.T. & Johnson, L. (1986) **Competent Expert Systems: a case study in fault diagnostics.** Kogan Page, London.

Martin, J. & McClure, C. (1985) **Diagramming Techniques for Analysts and Programmers.** Prentice Hall.

Winograd, T. (1983) **language as a Cognitive Process.** Addison Wesley, London.

# KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS

Brian R. Gaines

Application of expert systems is impeded by the knowledge engineering bottleneck. This has led to the development of rapid prototyping tools such as PLANET [10], ETS [1], MORE [7], SALT [8], MOLE [3], KITTEN [9], AQUINAS [2] and KSSO [5] which expedite knowledge acquisition and transfer [4,6]. These have proved practically useful but the range of applications of existing tools is limited, and it is not clear what limitations are inherent in the prototyping tools, what arise from the shells and their knowledge representation and inferencing procedures, and what arise from our lack of understanding of the underlying processes of knowledge acquisition and transfer. There has also been very little analysis of the developmental processes for knowledge-based system in relation to well-established techniques for system specification, rapid prototyping, evaluation, full-scale development, documentation and maintenance of conventional data-processing systems.

## FUNCTIONALITY REQUIRED IN EXPERT SYSTEMS

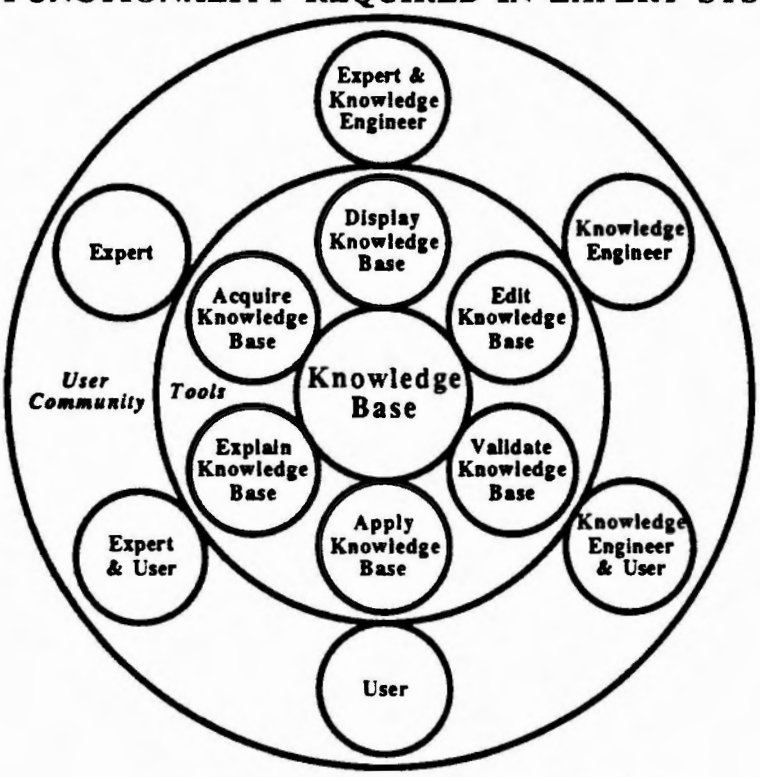



**Fig.1 Functionality of expert systems**

In considering the development of expert systems, it is important to realize that they are complex integrated systems involving a variety of tools and associated user environments, and that they also involve different categories of operator. Fig. 1 shows the functionality required in an expert system:

• *Knowledge base*: the central ring shows the knowledge base of facts and inference rules that is the heart of an expert system. This extends the facilities of a conventional database by enabling the storage not only of facts but also of rules that enable further facts to be derived. These rules may both be explicitly expressed (production rules) and implicit in the conceptual structure in which the facts are expressed (objects, frames).

• *Expert system shell*: the middle ring shows the expert system shell that provides computational facilities for applying the knowledge base to user decision support:

—the *application system* at the bottom tests the user's hypotheses or seeks to satisfy his goals by deriving the consequences of facts about a particular situation reported by the user, using the general facts and inference rules supplied by the expert and edited by the knowledge engineer;

—the *explanation system* to the lower left answers queries about the way in which facts have been derived in terms of what information and inference rules have been used;

—the *acquisition system* to the upper left provides tools for interviewing the expert to obtain his vocabulary, and general facts and inference procedures in terms of it;

—the *display system* at the top provides tools for presenting the knowledge base in an understandable form, and the relations between the facts and inference procedures it contains;

—the *edit system* to the upper right provides tools for editing the knowledge base while maintaining its integrity in terms of the vocabulary, variables and operations used;

—the *validation system* to the lower right provides tools for checking the knowledge base against specific case histories with known consequences.

*Dr Gaines is Director of the Knowledge Science Institute, University of Calgary, Calgary, Alberta, CANADA T2N 1N4*

• *Operators*: the outer ring shows the three classes of operator of expert systems:
—*users* come to the system for advice on specific problems in a particular domain, supplying it with specific facts and their hypotheses about consequences or goals;
—*experts* come to the system to transfer their knowledge about the problem domain, supplying it with generally applicable facts and inference procedures;
—*knowledge engineers* act as intermediaries between the expert and the system, helping him to encode his knowledge and validate the operation of the resultant expert system.

Most current expert system shells:
• offer a limited knowledge base represented in terms of production rules and frames;
• have weak application capabilities limited to deriving consequences and testing assertions;
• give simple explanations in terms of the facts and rules used;
• have no integrated acquisition capabilities;
• have limited capabilities to display the knowledge base, sometimes in graphical form;
• have interfaces to the knowledge base editor requiring understanding of internal data structures;
• have limited validation capabilities and no integrated database of case histories.

## THE VIRTUAL MACHINE HIERARCHY FOR EXPERT SYSTEM?




**Fig.2 The VM hierarchy underlying expert systems**

Knowledge-based systems are at the top of the *virtual machine hierarchy* which has developed in computing, and there are many layers of technology between an expert system and the computer on which it is running. In theory, system developers should not need to know about the lower levels of the hierarchy—machine architectures, operating systems and implementation languages are remote from knowledge processing. However, in practice, these lower levels are the foundations on which systems are built and any defects in them can undermine the functionality of the upper levels.

Figure 2 shows significant levels of the virtual machine hierarchy for knowledge-based systems:
• *Knowledge Acquisition Tools*: at the top of the hierarchy are the tools for automating the systems analysis for expert systems, through automatic interview procedures, modeling actual expert behavior, and analysis of knowledge in textual form.
• *Expert System Support Environment*: at the second level of the hierarchy is the equivalent of the Application Programming Support Environment (APSE) in conventional systems. The Expert System Support Environment (ESSE) has already been described in terms of Figure 1.
• *Expert System Shell*: at the third level of the hierarchy is the run-time system that elicits problem-specific information from the user, provides advice based on its knowledge base, and explains that advice in as much detail as required.

- *Expert System Shell Development Language*: at the fourth level of the hierarchy is the language in which the expert system shell is written, generally a special-purpose environment for coping with knowledge representation and inference.
- *Implementation Language*: at the fifth level of the hierarchy is the implementation language which actually interfaces to the computer. This tended to be Lisp in the early days of expert systems but as speed and space efficiency have become significant, and knowledge representation has become better understood, other languages that support dynamic data structures such as C and Pascal have become widely used.
- *Operating System*: at the sixth level of the hierarchy is the operating system within which the implementation runs. This needs to provide good interfaces to other programs, large databases and communications.
- *Machine Architecture*: at the lowest level of the hierarchy is the machine on which the expert system runs. This is really irrelevant, but use of Lisp in the early days of expert systems has focused attention on Lisp machines. Nowadays any good workstation is adequate.

## PROBLEMS IN EXPERT SYSTEM DEVELOPMENT

The preceding section have given an overview of expert systems and an indication of some of the problems that may arise in their development. Many of these relate to the knowledge engineering bottleneck but there are many others that are not direct outcomes, and the knowledge engineering problems in themselves have many facets.

There are what may be termed the problems of *professionalism* in establishing prototyping facilities:

- *Theory*: the lack of scientific and engineering foundations for knowledge-based systems is a major impediment to the development of professional standards for expert system development. We understand remarkably little about the nature of expertise even though it one of the most valued commodities in our society. We also have little fundamental understanding of knowledge, the ways in which it is represented in human thought processes, the ways in which it is used in human problem solving, and the ways in which it is transmitted between people.
- *Techniques*: the lack of well-established techniques for knowledge engineering is a major problem for new development teams. There is some folklore, generally mutually contradictory, a few rules of thumb based on experience, but so far no in-depth studies of what is actually done and how successful it has been.
- *Tools*: the lack of commercially available tools for knowledge acquisition and transfer is also a major problem. There are now many research reports of such tools but even those developed industrially have not yet been developed into products.
- *Training*: the training situation is improving all the time, largely because it has become the major secondary industry to the supply of expert system shells. A range of courses is being offered by commercial and university organizations. It is still a problem to get good quality courses targeted to specific requirements and on the schedules required.

Knowledge acquisition and transfer processes in human society are complex and poorly understood. Neurology, psychology, linguistics, education, sociology, anthropology, philosophy and systems theory all have significant contributions to make. However, integrating them, disambiguating jargon, and combining different objectives and perspectives are major problems.

There is a variety of short-term problems arising from these weaknesses in foundations:

- *System Specification*: the formal specification of software that has made software engineering a professional discipline which can ensure quality within well-defined budgets and schedules is notably lacking for expert system developments. We seem at the stage of conventional software development of twenty years ago. This should change as expert system developments move out of the laboratories into the hands of experienced software development groups trained in software management. There is little in the literature to support and expedite this change currently, and we need to develop specification disciplines for expert systems.
- *Access to Expertise*: the problems of getting experts to make the time commitments necessary to the development of expert systems should never be underestimated. Successful projects inevitably seem to have involved the experts in such a way that they become personally committed to the success of the system.
- *Acquisition of Skills*: communicating the basis of skilled performance between people, and between people and computers, is very difficult.

- *Representation of Knowledge*: encoding knowledge within the representation schemes of current shells can present major difficulties. Production rules as used in early expert systems offer very limited facilities for encoding simple knowledge structures at a single level. Frames and their generalization through objects allow conceptual hierarchies to be better expressed. Neither rules nor frames are adequate to encode the procedural or temporal knowledge which is often a key component of expertise.

Rapid prototyping is a significant practical objective and techniques for accelerated initial development without loss of long-term quality are now common in conventional data processing. However, expert system development emphasizes weaknesses in some areas of performance which we tend to take for granted in standard information technology. There are problems with:

- *Validation*: because we can rarely define and quantify expertise, the validation of expert systems can be very difficult. Do we subjectively validate against the opinions of the expert or do we attempt objective validation of performance against performance standards, accepting that the experts may.not themselves satisfy such criteria. How, do we determine whether the knowledge base captures the expert's relevant knowledge, accurately, consistently and completely? There are very few validation studies in the literature.
- *Usability*: the usability of expert systems raises many problems. If we accept the jargon of the expert and the rules of inference based on it, .how do we ensure that the operators will use this jargon in the same way as the expert so that the inferences remain valid. It is such considerations that make nonsense of the popular stereotype that access to encoded expertise can make experts of all of us. Understanding the questions, advice and explanations from an expert system generally requires an operator with the same professional background as the expert.
- *Maintainability*: the superficiality of the knowledge encoded in current expert systems raises major problems of maintenance. Keeping the knowledge base up to date may require an ongoing effort at the same level as the original development, a situation very different from most conventional data processing. The usual problems of extensive maintenance not introducing problems of its own are at least as severe for expert systems as for standard data processing.
- *Upgradability*: many expert systems, and the shells supporting them have been designed as stand-alone packages with little scope for integration into large-scale data processing environments. These systems will be difficult to upgrade and integrate with other software.

## REFERENCES


[1] Boose, J.H. (1986). Rapid acquisition and combination of knowledge from multiple experts in the same domain. **Future Computing Systems 1**(2), 191-216.

[2] Boose, J.H. & Bradshaw, J.M. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. **International Journal of Man-Machine Studies 26**(1) (January).

[3] Eshelman, L., Ehret, D., McDermott, J. & Tan, M. (1987). MOLE: A tenacious knowledge acquisition tool. Carnegie-Mellon University. **International Journal of Man-Machine Studies 26**(1) (January).

[4] Gaines, B.R. (1986). Foundations of knowledge engineering. Bramer, M.A., Ed. **Research and Development in Expert Systems III**. pp.13-24. Cambridge: University Press.

[5] Gaines, B.R. (1987). Rapid prototyping for expert systems. Oliff, M., Ed. **Proceedings of International Conference on Expert Systems and the Leading Edge in Production Planning and Control**. pp.213-241. University of South Carolina.

[6] Gaines, B.R. (1987). An overview of knowledge acquisition and transfer. **International Journal of Man-Machine Studies 26**(4) (April).

[7] Kahn, G., Nowlan, S. & McDermott, J. (1985). MORE: an intelligent knowledge acquisition tool. **Proceedings of the Ninth International Joint Conference on Artificial Intelligence**. pp. 581-584. California: Morgan Kaufmann.

[8] Marcus, S., McDermott, J. & Wang, T. (1985). Knowledge acquisition for constructive systems. **Proceedings of the Ninth International Joint Conference on Artificial Intelligence** pp. 637-639. California: Morgan Kaufmann.

[9] Shaw, M.L.G. & Gaines, B.R. (1986). Advances in interactive knowledge engineering. Bramer. M.A., Ed. **Research and Development in Expert Systems III**. pp.111-122. Cambridge: University Press.

[10] Shaw, M.L.G. & Gaines, B.R. (1986). Interactive elicitation of knowledge from experts. **Future Computing Systems, 1**(2), 151-190.

# KNOWLEDGE ACQUISITION AND MACHINE LEARNING - THE ISSUE OF MODELLING

Katharina Morik [1]

## 1 Knowledge Acquisition as Transfer

Knowledge acquisition and knowledge engineering has been viewed as a transfer process for a long time. This view is best illustrated by the bottleneck metaphor cited all over in the literature: knowledge acquisition is the bottleneck of building expert systems; knowledge engineering means extracting knowledge out of the expert and pushing a completed model into the expert system.
We can assign two interpretations to this metaphor. On the one hand, the expert is meant by "bottle". He has all the knowledge and the problem is to get it out. Knowledge elicitation techniques such as interview techniques, content analysis, and usage of repertory grids are psychological means to this end. An example of a system supporting knowledge elicitation is AQUINAS (Boose 87). The most important experience with knowledge elicitation is that the model to be represented in an expert system is not already fully specified in the head of the expert! Rather, the model comes into being during and because of the interaction between expert and elicitator (be it a system, or a man). The model to be represented in the expert system originates from the expert explicating her/his expertise. Thus, there is no transfer of an already completed model into a system, but a process of model building.
On the other hand, the bottleneck metaphor is interpreted identifying the expert system with the bottle. Systems supporting knowledge engineering serve as "funnels": knowledge editors, menu interfaces, knowledge base debuggers. However, an EMYCIN-like expert system's uniform representation of diverse types of knowledge, representation at the object level only, implicit representation of goals, methods, and strategies prevent the acquisition system from supporting the user in a focused and knowledgeable way (Clancey 86,Neches 85). Thus, the lesson learned from this approach is: tools cannot help the knowledge engineer more than the expert system's representation language allows them to. Therefore, newer developments in expert systems are motivated by the acquisition problem. In terms of the metaphor, the bottleneck is attempted to be widened, making a pot out of the bottle. An example for an acquisition systems based on this newer approach is MOLE (Eshelman 87). But still, this approach is based on the transfer view of building up a knowledge base. The model to be represented in the expert system is presupposed. The system supports the encoding of a given model but not the building of the model itself.

## 2 Knowledge Acquisition as Model Building

As opposed to transfer, modelling has neither a fixed start nor a fixed goal. Rather, some assumptions, layouts, and terminological definitions are taken as a starting point (first phase), a tentative model is built up (second phase), and the model is evaluated (third phase). Then, based on the result of the evaluation the starting point is revised, thus reopening the modelling cycle. Note that not only the model can be modified but also the terminology in which it

---

[1] Technical University Berlin, Institute for Applied Computer Science, Project Group KIT, Project KIT-Lerner; research is partially funded by the German Ministry of Research and Technology under contract ITW8501B1

is expressed. This process has much in common with scientific investigations: observations are to be structured and interpreted, hypotheses are established and used for predicting new observations, a theory is constructed out of the hypotheses, experiments are designed in order to validate the theory. Theories change in order to cover more phenomena or to cover phenomena in a simpler and more elegant way. As research, modelling is an infinite process and it requires reversability of all layouts and assumptions at all times.
Looking at knowledge acquisition this way we demand for systems, which support a user in her/his modelling, to provide for revising, reorganizing, and re-representing activities in a focused and transparent way. The demand for full reversability discriminates *step-wise restricting* knowledge acquisition from modelling. In step-wise restricting acquisition each definition restricts the space of further possible definitions, facts, and rules. It is impossible to retract or change a definition because of incoming new definitions or - even worse - new facts. Instead, the conflicting definitions or facts are rejected. Step-wise restriction has a sound logical basis. However, it forces the user to a *top down procedure* in acquisition: defining terminology from the most general to the most specific terms, then giving rules and facts which are controled by the terminology [2]. There is no system-supported way back from the more specific entries to the more general definitions. In practice, the user first models the domain using ink and paper or a text editor, then types in the domain model starting with the terminology. In case revisions are necessary, the knowledge base is loaded in an editor, revisions are made there by hand, and the edited knowledge base is loaded back into the system.
In addition to the top down procedure, a system supporting modelling allows for a *bottom up procedure* of acquisition: starting with well known facts, definitions are developed which control the facts appropriately. If a more specific definition contradicts a more general one, the user may decide which one has to be changed and the system revises the model accordingly. The problem with this procedure is to define the semantics of the revising operations and to compute all consequences of the changes. Up to now, there are no systems supporting the whole modelling cycle with full reversability.

## 3 Machine Learning as Automatic Modelling

In the field of machine learning, many aspects of modelling have been investigated and prototypical systems have already been built which perform some tasks of the modelling cycle:

- inducing classification rules from object descriptions, thus building up taxonomies which structure given data ('observations');
- discovering regularities,
  generating hypotheses, testing them, and using them in order to create a knowledge base with a considerable coverage;
- enhancing problem-solving performance by 'chunking' operators;
- learning by using analogy;
- revising a model because of conflicts, or failures, or master's advice;
- designing experiments.

The overall modelling cycle, however, has not yet been realized by any learning program.
As opposed to step-wise restricting acquisition systems, learning systems realize the bottom-up approach: terminology or, to put it in other words, explicating the intension of sets is created on the basis of facts, i.e. the extension of sets. Definitions are built up incrementally. They can be changed and enhanced because of new facts. This fits well to the demand for reversability of terminology. However, also automatic acquisition faces the problem of changing the representation, which was taken as a starting point - yet at another level.

---

[2] Terminology controls facts and rules in that it defines the semantics of the objects, attributes, and relations which are used to write facts and rules.

The examples have to be represented so that the learning program can use them to form the concepts, taxonomies, definitions, which in turn are used to express more facts and rules about the domain. Choosing the appropriate description language for the examples determines the possible outcome of the learning program. Revising the description language because of unsatisfactory learning results is a most challenging approach for meeting the reversability requirement (Amarel 68). Today, this task is either left to a teacher in most similarity-based learning-programs, or to the knowledge engineer who has built the knowledge-base, which is refined by explanation-based learning.

## 4 Integrating Machine Learning into Knowledge Acquisition

Viewing machine learning as automatic modelling and knowledge acquisition as a modelling task, it is an obvious idea to have machines perform parts of the task of acquisition. Machine learning has been successfully applied to building large parts of knowledge bases. Examples of such successful learning programs are ID3 (Quinlan 83) and AQ11 or INDUCE (Michalski 80,83). Most often the learning system and the performance system are linked *serially* : The learning system acquires rules independently from the knowledge which is already stored in the expert system. It delivers rules (or facts, or tables) to the performance system without getting or using any feedback from it. Another way of putting learning programs to good use in knowledge acquisition is shown by apprenticeship-learning, where the learning program *interacts* with the expert system. It compares the performance system's solution with a solution given by an expert, explains the difference and changes the knowledge base accordingly. An example of such a system is LEAP (Mitchell 85). However, this type of learning is restricted to the end of the second phase of the modelling cycle.
A third way to integrate machine learning into knowledge acquisition is *balanced cooperative modelling* . A knowledge-acquisition system supports the user in her/his modelling and at the same time enhances this model by learning. BLIP [3] is a system which supports the modelling activity of the user through the first two phases by a convenient interface, a coordinator of different knowledge sources, and by watching consistency and integrity. The user may well follow the bottom-up procedure or the top-down procedure of acquisition. Also terminological knowledge can be changed; the system manages the consequences of the revisions. BLIP is also an actively modelling system which is designed to enhance the model by learning and revising. Moreover, the same procedures which integrate the new facts, rules, or objects given by the user also incorporate those given by another part of the system, the learning component. Machine learning techniques form new concepts and find new rules as does the user. Other modules derive new facts from the rules, detect contradictions and handle them as does the user if s/he discovers them. Thus, both user and system (re-)build, (re-)structure, and (re-)organize the model.

## REFERENCES

Amarel, S.(68): "On Representation of Problems of Reasoning about Actions"; In: B. Meltzer/ D. Michie (eds.): Maschine Intelligence, Vol. 3, University of Edinburgh Press, p. 131-171, 1968

Boose, J.H., Bradshaw, J.M. (87): "Expertise Transfer and Complex Problems Using AQUINAS as a Knowledge Acquisition Workbench for Expert Systems"; In: International Journal of Man-Machine Studies (in press)

Clancey, W.J. (86): "From GUIDON to NEOMYCIN and HERACLES"; In: The AI Magazine, Vol.2, No. 3, pp. 40 - 60

---

[3] BLIP is currently under development at the TU Berlin, project KIT-Lerner

Eshelman, L., Ehret, D., McDermott, J., Tan, M. (87): "MOLE: A Tenacious Knowledge Acquisition Tool"; In: International Journal of Man-Machine Studies (in press)

Michalski, R.S./Chilausky, R.L.(80): "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis"; International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, 1980

Michalski, R.S./Stepp, R.E.(83): "Learning from Observation: Conceptual Clustering" In: Michalski/Mitchell/Carbonell (eds.): Machine Learning; Tioga Press, Palo Alto, 1983

Mitchell,T.M.; Mahadevan,S.; Steinberg,L.I. (85): "LEAP: A Learning Apprentice for VLSI Design"; In: Proc. IJCAI-85, Los Angeles 1985

Morik, K.J. (87): "Acquiring Domain Models"; In: International Journal of Man-Machine Studies (in press)

Neches, R./Swartout, W.R./Moore, J.(85): "Enhanced Maintenance and Explanation of Expert Systems through Explicit Models of their Development"; In: IEEE Transactions on Software Engineering, November, 1985

Quinlan, R.J. (83): "Learning Efficient Classification Procedures and their Application to Chess End Games"; In: Michalski/Mitchell/Carbonell (eds.): Machine Learning; Tioga Press, Palo Alto, p. 463 - 482

# MACHINE LEARNING AND EXPLANATIONS

Yves Kodratoff, Equipe Inference et Apprentissage,
LRI, Bat 490, Universite Paris-Sud

## Introduction

It became popular during the last few years to split ML in two parts.

One part has been called "Similarity-Based Learning" (but tends presently to be called *Empirical Learning*, and we shall use this last way of speaking in the following). It covers all programs

- that rely on an extensive use of several examples even if not hundreds of them, that analyse ressemblances and differences among the (positive and/or negative) different examples,
- that rely more on inductive hypotheses than on deduction. Contrary to what can be believed at first sight these programs do need as much as possible of domain deep background knowledge. They use this deep knowledge to perform better grounded induction and rely on it for validating the acquired knowledge.

We want to insist on this last point since beginners in ML often believe that Empirical Learning is precisely characterized by a lack of use of deep knowledge. It is quite true that one can learn from sets of positive and negative examples by using only the implicit background knowledge contained in the representation they are given. Nevertheless, as soon as changes in representation are needed, even very simple ones, they can only be generated from stored background knowledge unless one can accept random changes, which is very seldom the case.

The other has been called *Explanation-Based Learning* (EBL) and tends nowadays to be put under the more general heading of *Analytical Learning*. It covers all programs

- that rely on an extensive use of background knowledge, that "exploit the intra-example relationships, which constrain the interpretive or explanatory concepts applied to one or more facts or examples" (Michalski 1986), i.e. that analyse ressemblances and differences inside each example,
- that rely more on deduction than on inductive hypotheses.

We have now to insist on the reverse point as for Empirical Learning. One will be tempted to characterize EBL by the fact that it is purely deductive and uses no induction. This would be false as well. As we have said, Emprical Learning uses deduction to control the inductions done from the examples. On the contrary, EBL uses deduction from background knowledge to provide a *justification* that can be also called a *proof trace* of the example it is analysing. Generalization of the proof trace proceeds by induction which may be controlled by *operationality criteria*, as advocated by Mitchell (Mitchell et al. 1986), or by *pre-stored schemes* (i.e. an other kind of background knowledge), as advocated by DeJong (DeJong et al. 1986). Such a generalization is what is called an *explanation*.

This should have make quite clear two points. On the one hand, Empirical Learning and EBL represent quite different approaches to ML:

- one uses *inter*-examples while the other *intra*-example analysis,
- they have very different relationships towards deduction and induction,
- they use background knowledge in a very different way.

On the other hand, their differences do not stem from the fact one would use induction only without background knowledge, and that the other would only deduce from background knowledge without inducing.

These two different approaches bring in different kinds of knowledge. It is therefore not surprising that very soon, we could see appear systems that use both of them.

## 1 - Explanations in ML

Let us first notice, with M. Ebel (Ebel 1981), that it is surprizing to discover how much has been written on argumentation, and how little on explanation. One could explain this fact by holding that an argumentation is simply an explanation given to answer implicit questions, which is quite true but not precise enough. This why, before starting a discussion about explanations, especially on what they are and what is their role in ML, let us first make clear the differences between the somewhat related concepts of *explanation*, *proof*, *justification*, and *argumentation*.

### Definitions

As pointed out by Braithwaite (Braithwaite 1968), "Molière was right in laughing at the doctors who offered the *virtus dormitiva* of opium as the answer to the question as why opium produced sleep. But it would not be foolish to answer the question as to why a particular specimen of powder produced sleep by replying that it was because that powder was opium and that opium had the property of producing sleep", i.e. it has a virtus dormitiva. This example shows very clearly that *an explanation is an answer to a question* (we shall refine this defintion later) since, depending on the question, viz. "Why opium produces sleep?" or "Why this powder produces sleep?", the same answer, viz. "Because it has a virtus dormitiva" is ridiculous or fully explanatory.

*Argumentation*, is classically defined (see for instance Perelman 1958) as being a *technique of the discourse allowing to increase the acceptation of one's thesis by the other's minds.* This means that *scientific argumentation* is nothing but *a formal proof using premisses acknowledged as true by the ones to be convinced.* This means that an argument is not any kind of proof, but the one which uses special premisses that are already accepted by those towards which the argument is aimed at.

As noticed by Meyer (Meyer 1975) the definition of an explanation given by Hempel should actually considered as the good one for a justification. This is why we shall define, slightly distorting Hempel's definitions (Hempel 1965), a *justification* as being formed of a *Justifying Part* (our transcription of Hempel's *Explanans* ) from which is *logically deduced* a *Justified Part* (our transcription of Hempel's *Explanandum*).

The *Justifying Part* consists of a *set of true antecedant conditions*, and of a *set of valid general laws*. For instance, in order to explain the cracking of an automobile radiator the set of antecedant conditions is relative to the fact the car was left outside, that the temperature dropped down, that the radiator is made of such metal, that it contains such fluid, and all other information specific to the case under study. In our ML vocabulary, we speak of this as *the knowledge relative to the training example*. The set of valid laws would contain "laws" such as the following: the freezing temperature of water, the dilatation coefficients of the metal and of ice, laws of increase in pressure when volume is limited, and all other information that is relevant, but not specific to the case under study. In our ML vocabulary, we speak of this as the *Domain Theory*. Both of them constitute a *Justifying Part*

The *Justified Part* is the *description of the empirical phenomenon to be explained.* In our ML vocabulary, we speak of this as *the training example itself.*

The validity of the justification follows when one can formally prove that the justifying part implies the justified part. It is then clear that a complete scientific justification is a prediction as well, as pointed out by Hempel. In our ML vocabulary, we would say that the justifying part is the set of asserted data and of clauses that will be necessary to carry out the proof of the justified part. In other words, it contains the same information as the trace of the proof one has to perform to validate the training example.

"The 'Why?' question is not expected to be answered by detailing all the events which together make up a total cause ... all is usually expected is the part-cause which is of most interest to the questioner - which presumably is that of which he is ignorant" (Braithwaite 1968). Therefore, a real explanation must have also an argumentative value. We can define an *explanation as a justification which is also an argument.* In our ML vocabulary this means that *an explanation is a modification of a proof trace the argumentative value has been improved* by deleting the non-relevant information (what is named in (Mitchell et al. 1986) the application of the operationality criterion), by generalizing either some constants into variables as in (Mitchell et al. 1986), by transforming some predicates into more general ones or into more particular ones when the proof has been done to a too high level of generality, as advocated in (DeJong et al. 1986).

## 2 - A typology of Explanations in ML

A more detailed account will be presented in Appendix one. We give here only their taxonomy, more details are given in (Kodratoff 1987).

2.1 - **(KNOWLEDGE) REPRESENTATION OF EXPLANATIONS**
- 2.1.1 - **Shank's XPs** (Shank 1987, 1988)
- 2.1.2 - **Explanations in PROTOS** (Porter and al. 1986, 1988)
- 2.1.3 - **Justifications in DISCIPLE** (Kodratoff and al. 1987a)

2.2 - **EXPLANATION SYNTHESIS**
- 2.2.1 - **In hard Theories**
  - 2.2.1.1 - Explanation-Based Learning, Explanation-Based Generalization (Mitchell and al. 1986) (DeJong and al. 1986)
  - 2.2.1.2 - Context Recognition (Costa and al. 1987)
- 2.2.2 - **In soft Theories**
  - 2.2.2.1 - DISCIPLE's explanations (Kodratoff and al. 1987a)
  - 2.2.2.1 - Analogy-Based Explanations (Kodratoff and al. 1987b)
- 2.2.3 - **Constructive Explanations** (Biermann and al. 1984) (Franova 1985, 1986)

## REFERENCES


(Biermann and al. 1984) Biermann A. W., Guiho G., Kodratoff Y., *Automatic Program Construction Techniques*, MacMillan Publishing Company, New York 1984.

(Braithwaite 1968) R.B. Braithwaite, *Scientific Explanation*, Cambridge University press, 1968.

(Costa and al. 1987) Costa E. , Duchénoy S. , Kodratoff Y. in "Intelligent Computer Aided Instruction" "A resolution based method for descovering students'misconceptions.", Self J. (ed) , Chapman and Hall, January 1987.

(Dejong and al. 1986) Dejong G., Mooney R. "Explanation-Based Learning: An Alternative View", Machine Learning 1, 145-175, 1986.

(Ebel 1981) Ebel M. "L'explication: acte de langage et légitimité du discours", Revue Européenne des Sciences Sociales 19, n°56, 1981, 15-36.

(Franova 1985) Franova M., "CM-Strategy: A Methodology for Inductive Theorem Proving or Constructive Well-Generalized Proofs", Proc. IJCAI-85, Los Angeles 1985, pp. 1214-1220.

(Franova 1986) Franova M., "Proving Theorems by Construction of Valid Formulae", in Information Processing 86, Proc. IFIP'86, Kluger H.G. ed., North Holland 1986, pp. 41-46.

(Hempel 1965) C.G. Hempel, *Aspects of Scientific Explanation*, The Free Press, New York 1965

(Kodratoff and Ganascia 1986) Kodratoff Y., Ganascia J.G., "Improving the generalization step in Learning" in Machine Learning: An Artificial Intelligence Approach, Volume II, R.S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), Morgan Kaufmann 1986, pp 215-244.

(Kodratoff 1987) Kodratoff Y., "Is AI a sub-field of Computer Science - or is AI the Science of Explanations?", in Progress in Machine Learning, I. Bratko and N. Lavrac eds, SIGMA Press, Wilmslow 1987, pp. 91-106.

(Kodratoff and al. 1987a) Kodratoff Y., Tecuci G. "Techniques of design and DISCIPLE Learning Apprentice", International J. of Expert Systems 1, 1, 1987, 39-66.

(Kodratoff and al. 1987b) Kodratoff Y., Vrain C., "Analogy and Dissimilarities", in KROML Proceedings, K. Morik eds, Springer Verlag LNCS series, Berlin 1987.

(Michalski 1986) Michalski R.S., "Understanding the Nature of Learning: Issues and Research Directions", in *Machine Learning: An Artificial Intelligence Approach, Volume 2*, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Morgan Kaufmann 1986, pp. 3-25.

(Michalski 1983) Michalski R.S., "A Theory and Methodology of Inductive Learning" in Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Morgan Kaufmann 1983, pp 83-134.

(Mitchell and al. 1986) Mitchell T. M., Keller R. M., Kedar-Cabelli S. T. "Explanation-Based Generalizations: A Unifying View", Machine Learning 1, 47-80, 1986.

(Perelman 1958) Perelman Ch., Olbrechts-Tyteca L., *La nouvelle rhétorique. Traité de l'argumentation*, Presses Universitaires de France, Paris 1958.

(Porter 1986, 1988) Porter B. W., Bareiss E.R. "PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks", in Proc. IMAL , Les Arcs 1986, pp. 159-174. To appear also in *Machine Learning: An Artificial Intelligence Approach, Volume 3*, Morgan Kaufmann 1988.

(Shank 1987) Shank R. C. Explanation Patterns : Understanding mechanically and Creatively, Ablex Publishing Company, 1987.

(Shank 1988) Shank R. C., Kass A. "Explanations, Machine Learning, and Creeatiivity", to appear in *Machine Learning: An Artificial Intelligence Approach, Volume 3*, Morgan Kaufmann 1988.

## APPENDIX: A TYPOLOGY OF EXPLANATIONS BORN FROM AI

### 1 - (KNOWLEDGE) REPRESENTATION OF EXPLANATIONS

#### 1.1 - Shank's XPs

We shall use the example of the assassination of the French king Henri IV by Ravaillac.
An XP is a kind of script for an explanation. It is made of 4 parts :

- an index (or name) that allows its retrieval

*scenario1: LOVE-MURDER of x by y*

*scenario2: REVOLUTIONARY-MURDER of x by y*

*scenario3: . . .*

- a set of states that activate the XP (preconditions)

*scenario1: x et y love the same person z, x knows, y does not, etc ...*

*scenario2: the social position of x is much higher than the one of y, society is in a turmoil, ...*

In the case of scenario1 the preconditions are not met, therefore in actual use this scenario would be rejected at once.

- a script describing the causal links of the sequence of events

*scenario1: x AND z are showing each other their feelings, y unexpectedly sees them (this explains his/her wrath), ...*

*scenario2: x bears high responsabilities, y does not (this explains why x and y are not usually together). y tries to find ways to come near x ...*

- the resulting states (post- conditions) of the scenario

*scenario1: y is tried. Light sentence ...*

*scenario2: y is not at once punished. Often, he/she even gets x's estate ...*

In the case of scenario2 the postconditions are not met, therefore in actual use this scenario would be rejected after all this testing.
This approach raises several problems

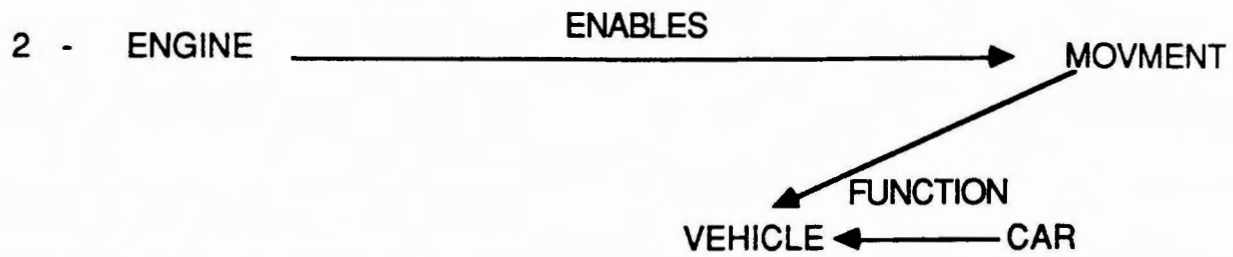
- Retrieval of the good scenario, that can be lenghty as the axample of scenario2 shows.
- "Tweaking" when applied to real cases, which implies efficient analogy procedures.
- hard to use in unexpected situations

#### 1.2 - Explanations in PROTOS

These methods are inspired by PROTOS (Porter 1986, 1988) in which the production rules are associated with **elementary explanations**, that describe what kind of relation holds between the antecedent and the consequent of each production rule.
There are 6 such elementary explanations. Among them, one finds the classical *ISA* ( generality relationship ), and *PART-OF* ( part to whole mappings ), but also *ENABLES* telling that a feature enables the function of an object ( e.g., wings enable flight ), etc ...
An **explanation** is a combination of such elementary explanations, and is given by a path in the relationship frame among objects.
From these explanations, one induces a recognition function that uses each property as a characteristic one.
*For example, consider the following two different semantic nets of relationships between* ENGINE *and* CAR.



Associated recognition function : x? is an engine if it is the part of a car.




Associated recognition function : x? is an engine if it enables the movment of any kind of vehicle.
In this method, it is part of the training provided by the teacher to underline the explanations that are significant. The system must still make explicit the recognition function associated with each explanation.

### 1.3 - **Justifications in DISCIPLE**

We shall describe them on the following example.
Suppose that the goal of the system is the following.

GOAL: ATTACH sectors ON chassis

that the system fails to realise this goal. It will ask the corrrect answer to its user. Suppose that the user answers as follows.

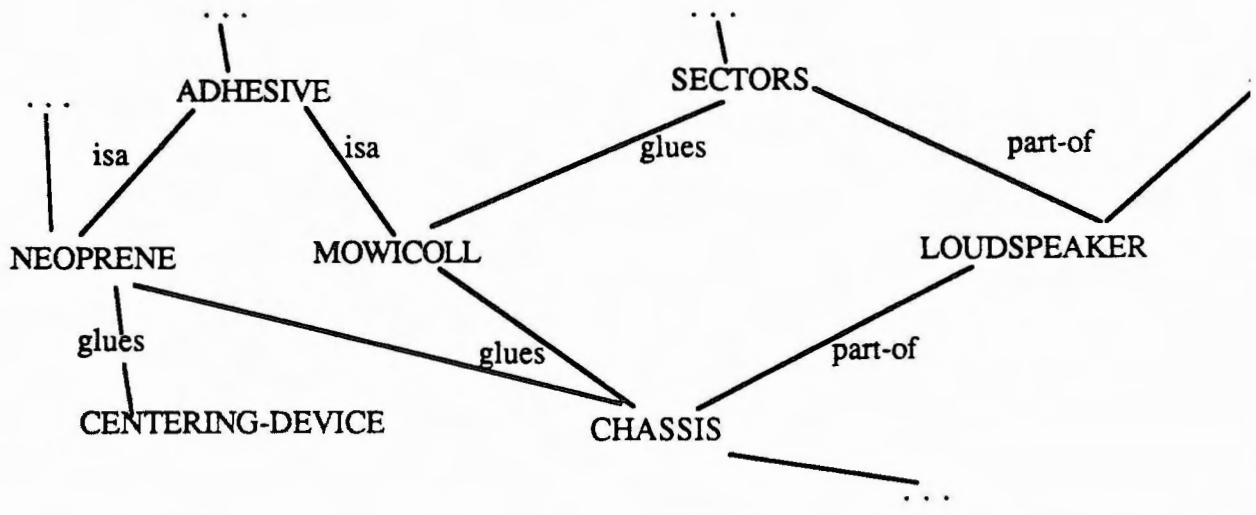
USER's ANWER:
APPLY mowicoll ON sectors,
PRESS sectors ON chassis.

This is an example of the RULE:

ATTACH sectors ON chassis
I--
APPLY mowicoll ON sectors,
PRESS sectors ON chassis.

The system will now ask the user why he proposed such a solution, referring to its semantic net. In our example let us suppose that the system knows the following.

**(uncomplete) semantic net**




Using this semantic net, the system can ask the following questions to its user (whose answers are underlined).

Does your solution works "because":
mowicoll GLUES sectors? yes
mowicoll GLUES chassis? yes
sectors PART-OF loudspeaker? no
chassis PART-OF loudspeaker? no

The positive answer of the user are then used as "justifications" for the rule he has been proposed: 'mowicoll GLUES sectors', 'mowicoll GLUES chassis' are such justifications.

:om this, the system deduces the rule:

IF

(mowicoll GLUES sectors) and (mowicoll GLUES chassis)

THEN

ATTACH sectors ON chassis

I--

APPLY mowicoll ON sectors,

PRESS sectors ON chassis.

## 2 - EXPLANATION SYNTHESIS

### 2.1 - In hard Theories

#### 2.1.1 - Explanation-Based Learning, Explanation-Based Generalization

Quite recently, (Mitchell and al. 1986) introduced a technique which, though recent, is already unambiguous and can be implemented straight away.

This method starts from a proof trace and generalizes it into an explanation. It is lengthy to describe, this is why we will now refer to the next problem, not yet resolved but easier to illustrate. A detailed description of this method will be given, in an other context, into the next issue of this review.

The obtaining of an explanation in a domain that shows a strong theory relies on the preliminary obtaining of a proof trace. To achieve this goal we need three informations

1 - a theory of the domain, given by the set of theorems that are valid in the domain.

*For example, in the domain of classical elementary logical riddles, one has :*

```
DOMAIN THEORY
IF   MAN(x)              THEN   MORTAL(x)
     IF   WOMAN(x)       THEN   MORTAL(x)
     IF   MAN(x)                THEN   HUMAN(x)
     IF   WOMAN(x)              THEN   HUMAN(x)
IF   MORTAL(x)           THEN   FALLIBLE(x)
                         ...
     IF   CRETAN(x)             THEN   LIAR(x)

                    ...
```

2 - a set of asserted data, given by instantiated tautologies.

*For example, in the same domain , one may have :*

```
                DATA BASE
MAN(SOCRATES)
AGE(SOCRATES)  = 43
WEIGHT(SOCRATES)  = 75
     GREEK(SOCRATES)
```

3 - a question, or a problem to solve, the proof of which will generate the required trace.

*Still in the same domain , one may ask :*

```
                QUESTION
MORTAL(SOCRATES)  ?
```

Answering a question is equivalent to proving the validity of the information it contains. This provides a proof trace.

*One obtains the trace :*

```
                PROOF TRACE
MAN(SOCRATES) ?
     YES
```

In this example, the proof trace is quite short, but this should not hide the fact that usual proofs may have many steps.

*Explained in "English" this proof trace constitute a trivial explanation of the reason why Socrates is mortal :*

***MORTAL(SOCRATES) IS TRUE BECAUSE MAN(SOCRATES) IS TRUE***

More generally, straightforward proof traces constitute trivial explanations. The problem, in order to get more interesting ones, is to obtain a generalization of these traces.

In our example, one may take into account the fact that *MAN* implies *HUMAN*, and generalize the explanation to

***MORTAL(SOCRATES) IS TRUE BECAUSE HUMAN(SOCRATES) IS TRUE***

There exist at present little indications on the way this generalization can be safely performed. Heuristics should be soon devised, as shown in (Dejong and al. 1986)

We are now in position to better explain what EBG does. Given a proof trace, all its expressions are fully instantiated. EBG is a technique that allows to generalize some of the constants found in the trace into variables or functional expressions. Since MAN is a predicate, EBG will be unable to generalize it to the other predicate HUMAN.

In case of longer proof traces, EBG generalizes them, without changing the names of the predicates, but by introducing some variables in place of the constants.

**What is learned from an explanation by the system?**

The explanation itself is a new rule that expresses usable knowledge. In general, explanations are a good way for the transmission of deep knowledge to shallow knowledge. The importance of this last problem is well-known but its link with EBG is not yet common knowledge. Our example will now allow us to illustrate our claim.

*From the explanation telling that Socrates is mortal because he is a man, one finds the rule* **IF** **MAN(SOCRATES)** **THEN** **MORTAL(SOCRATES)**

*which, by EBG will be generalized to*

**IF** **MAN(x)** **THEN** **MORTAL(x)**

*that is : one finds again an already known rule, and nothing is learned.*

Now, the reader can imagine that in nearer to real-life proofs, several inference steps will be done, and the generalized explanation will usually be different from any of the rules existing in the data base. These steps will combine informations necessary to the proof of the example. In other words, as our example shows, the explanation contains the information necessary for answering the question. This information is, so to say, singled out by the question. It is the shallow knowledge necessary to answer this particular question.

*Imagine now that one has been able to find the explanation that Socrates is mortal because he is human. One will therefore obtain the rule*

**IF** **HUMAN(SOCRATES)** **THEN** **MORTAL(SOCRATES)**

*which, through EBG, will generalize to*

**IF** **HUMAN(x)** **THEN** **MORTAL(x).**

*This is a new rule, different from any of the rules existing in the data base, expressing some shallow knowledge about mortality, and singled out by our particular question relative to Socrates.*

### 2.1.2 - Context Recognition

## 2.2 - In soft Theories

When the domain theory is weak, formal proofs, such as those needed by EBG, are no longer available. It may even happen that, forcing the expert to express him/herself in a way allowing such formal proofs, will prevent the expert delivering the correct knowledge.

In that case, the automation of knowledge acquisition can only be done under a constant expert supervision.

The system may use any of the above learning techniques, it will do its best by proposing generalizations, or explanations to the expert, who will be in charge to validate them.

This approach has been adopted in our system DISCIPLE (Kodratoff and al. 1987a).

### 2.2.1 - DISCIPLE

Recall that at the justification stage the system have been learning the following rule.

IF

(mowicoll GLUES sectors) and (mowicoll GLUES chassis)

THEN

ATTACH sectors ON chassis

I--

APPLY mowicoll ON sectors,

PRESS sectors ON chassis.

DISCIPLE will now over-generalize this rule simply by turning all its constants into variables.

RULE INDUCED by DISCIPLE:

IF

(z GLUES x) and (z GLUES y)

THEN

ATTACH x ON y

I--

APPLY z ON x,
PRESS x ON y.

This rule is certainly too general. To check this, DISCIPLE looks into its semantic net for other items that can be instances of the over-generalized rule. In our example, it will find one, and ask the user if it is a valid rule.

QUESTION ASKED TO THE USER: IS THE FOLLOWING RULE A VALID ONE?

ATTACH centering-device ON chassis
I--
APPLY neoprene ON centering-device,
PRESS centering-device ON chassis. ?

Suppose that the user approves of this rule, DISCIPLE will use classical Leaning from Examples techniques in order to find a least common generalization of the approved rules. In our example, this would lead to

IF
(z GLUES x) and (z GLUES y) and (ADHESIVE z)
THEN
ATTACH x ON y
I--
APPLY z ON x,
PRESS x ON y.

In case the user rejects a rule, it also used as a negative example by classical Learning from Example techniques.

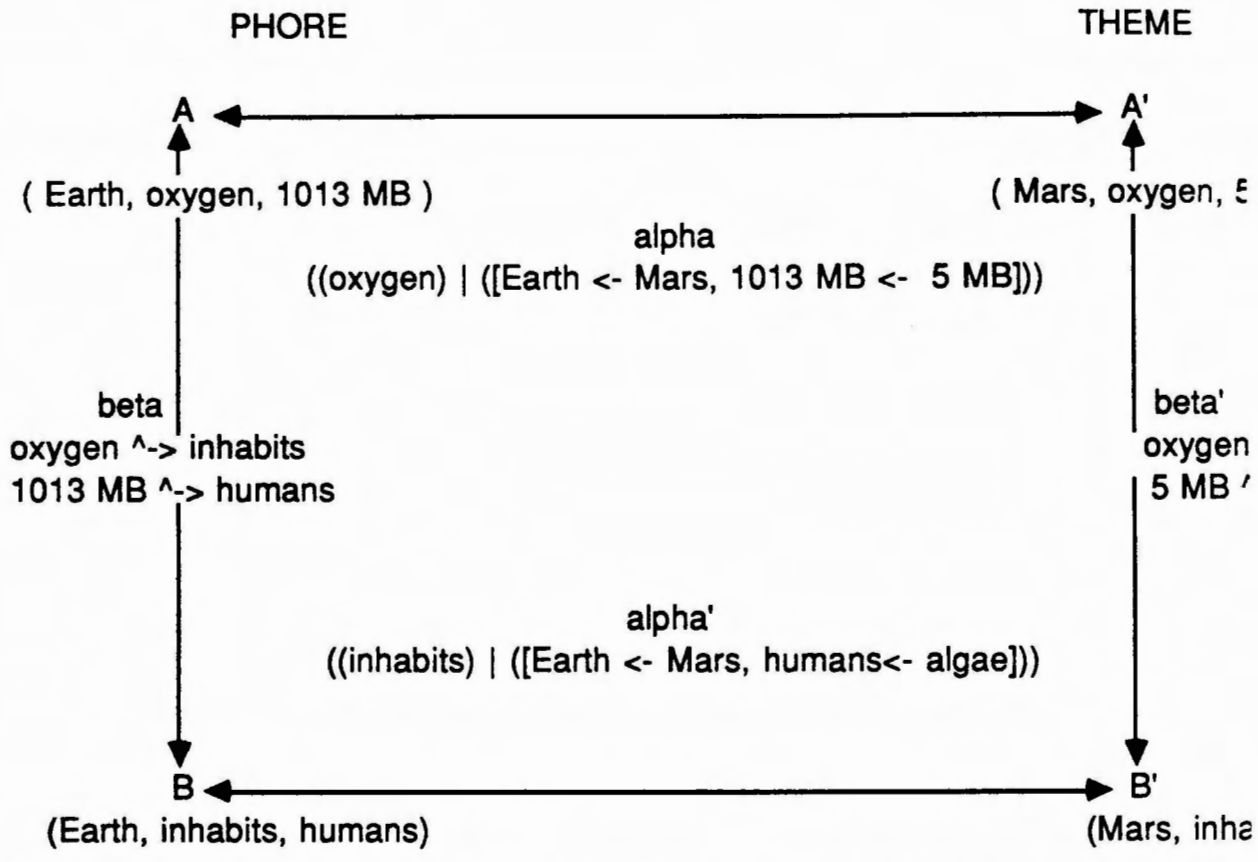
### 2.2.2 - Analogy-Based Explanations

Let us call phore ( holder ): {A , B}, the information that bears the analogy.
For example

A = Earth has an atmosphere of 1013 MB containing oxygen
B = Earth is inhabited by humans

Let us call theme: {A', B'}, the result of the analogy

A' = Mars has an atmosphere of 5 MB containing oxygen
B' = Mars is inhabited by algae




Suppose that between A and B there exists a causality relation b and that between A' and B' there exists a causality relation b '.
Suppose that between A and A' there exists a similarity relation b and that between B and B' there exists a similarity relation b '.
In our example, one has:

similarity relation between A and A' :

A and A' atmospheres contain oxygen , but in A, the pressure of the atmosphere is 1013 MB, while in A', it is 5 MB.

similarity relation between B and B' :

B and B' are both "inhabited", but B is inhabited by humans and B' is inhabited by algae.

LET US NOW SUPPOSE THAT THERE IS A CAUSAL RELATIONSHIP BETWEEN (OXYGEN + PRESSURE VALUE) AND THE TYPE OF LIFE PRESENT ON A PLANET THEN

THE OXYGEN EXPLAINS THE PRESENCE OF LIFE
THE DIFFERENCE OF PRESSURE EXPLAINS THE DIFFERENCE OF LIFE FORM

ONE CAN THEREFORE CONCLUDE THAT:

EARTH IS INHABITED BY HUMANS BECAUSE ITS ATMOSPHERE CONTAINS OXYGEN AND ITS PRESSURE IS 1013 MB.
MARS IS INHABITED BY ALGAE BECAUSE ITS ATMOSPHERE CONTAINS OXYGEN AND ITS PRESSURE IS 5 MB.

### 2.3 - Constructive Explanations

They mau be obtained from constructive proofs of existential theorems as does the system PRECOMAS (Franova 1986). It builds an instance in order to prove an existential theorem. This provides athe kind of explanations: here is an instance that proves your theorem.

# AQUINAS: A KNOWLEDGE ACQUISITION WORKBENCH FOR BUILDING KNOWLEDGE-BASED SYSTEMS

John H. Boose and Jeffrey M. Bradshaw
Knowledge Systems Laboratory, Boeing Advanced Technology Center, 7L-64
Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124

## Abstract

Acquiring knowledge from a human expert is a major problem when building a knowledge-based system. Aquinas, an expanded version of the Expertise Transfer System (ETS), is a knowledge acquisition workbench that combines ideas from psychology and knowledge-based systems research to support knowledge acquisition tasks. These tasks include eliciting distinctions, decomposing problems, combining uncertain information, incremental testing, integration of data types, automatic expansion and refinement of the knowledge base, use of multiple sources of knowledge, and providing process guidance. Aquinas interviews experts and helps them analyze, test, and refine the knowledge base.

Expertise from multiple experts or other knowledge sources can be represented and used separately or combined. Results from user consultations are derived from information propagated through hierarchies. By systematically sweeping through values of traits and observing the effects on test sets, Aquinas can optimize trait values and even suggest new classes of traits. Aquinas can deliver knowledge to other environments by creating knowledge bases for several different expert system shells. Help is given to the expert by a dialog manager that embodies knowledge acquisition heuristics.

Aquinas contains many techniques and tools for knowledge acquisition; the techniques combine to make it a powerful testbed for rapidly prototyping portions of many kinds of complex knowledge-based systems.

## Aquinas System Description

Aquinas, an expanded version of the Expertise Transfer System (ETS; Boose, 1984, 1985, 1986a), is a knowledge acquisition workbench that combines ideas from psychology and knowledge-based systems research to support knowledge acquisition tasks. These tasks include eliciting distinctions, decomposing problems, combining uncertain information, incremental testing, integration of data types, automatic expansion and refinement of the knowledge base, use of multiple sources of knowledge, and providing process guidance (Boose and Bradshaw, 1987). Aquinas interviews experts and helps them analyze, test, and refine the knowledge base. Expertise from multiple experts or other knowledge sources can be represented and used separately or combined. Results from user consultations are derived from information propagated through hierarchies. Aquinas delivers knowledge by creating knowledge bases for several different expert system shells. Help is given to the expert by a dialog manager that embodies knowledge acquisition heuristics.

Using Aquinas, rapid prototypes of knowledge-based system can be built in as little as one hour, even when the expert has little understanding of knowledge-based systems or has no prior training in the use of the tool. The interviewing methods in Aquinas are derived from George Kelly's Personal Construct Psychology and related work (Kelly, 1955; Shaw and Gaines, 1987). Kelly's methods and theory provide a rich framework for modeling the qualitative and quantitative distinctions inherent in an expert's problem-solving knowledge.

Aquinas tools mentioned mentioned in this paper are explained more fully in Boose and Bradshaw (1987). Aquinas is written in Interlisp and runs on the Xerox family of Lisp machines. Subsets of Aquinas also run in an Interlisp version on the DEC Vax and a micro computer "C/UNIX"-based portable version.

**Knowledge Represented in Rating grids.** The expert enters and refines knowledge in the form of *rating grids*. In a rating grid, problem solutions - *elements* - are elicited and placed across the grid as column labels, and traits of these solutions - *constructs* - are listed alongside the rows of the grid (Figure 1). Traits are first elicited by presenting groups of solutions and asking the expert to discriminate among them. Following this, the expert gives each solution a rating showing where it falls on the trait scale, and gives the relative importance of each trait.

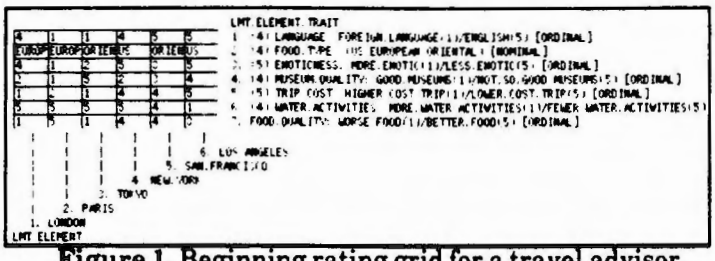



**Figure 1.** Beginning rating grid for a travel advisor.

After an initial grid is constructed, Aquinas helps the expert refine and expand the knowledge base by invoking a variety of analysis tools. Similarities between elements and traits are analyzed to help the expert refine useful distinctions and eliminate those that are inconsequential or redundant. Inductive implications between trait values can show the expert higher levels of abstraction implied by a rating grid. If the expert disagrees with an implication, Aquinas helps the expert refine the grid. Distinctions captured in grids can be converted to other representations such as production rules, fuzzy sets, or networks of frames.

**Hierarchical Rating Grids.** Using a major extension to Kelly's methods, experts use Aquinas to arrange rating grid knowledge in hierarchies and lattices. Hierarchical tools in Aquinas help the expert build, edit, and analyze knowledge in hierarchies. These hierarchies allow the expert to break up complex problems into pieces of convenient size and similar levels of abstraction. Hierarchies in Aquinas are organized around solutions, traits, knowledge sources (i.e., experts), and cases (Figures 2 and 3).

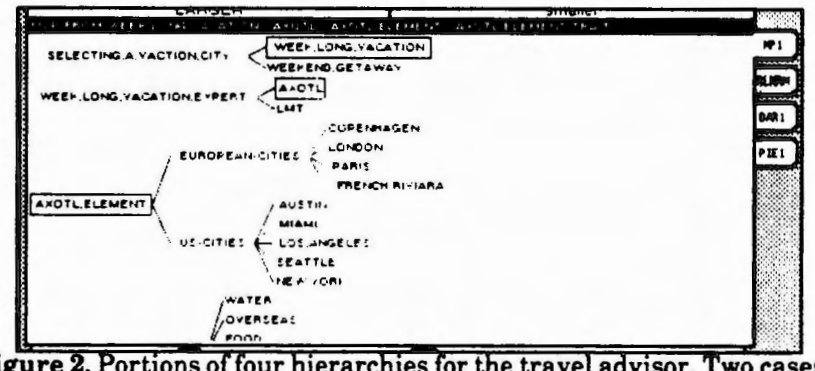



**Figure 2.** Portions of four hierarchies for the travel advisor. Two cases and two experts are shown, as well as expert Axotl's solution hierarchy for the WEEK.LONG.VACATION case.

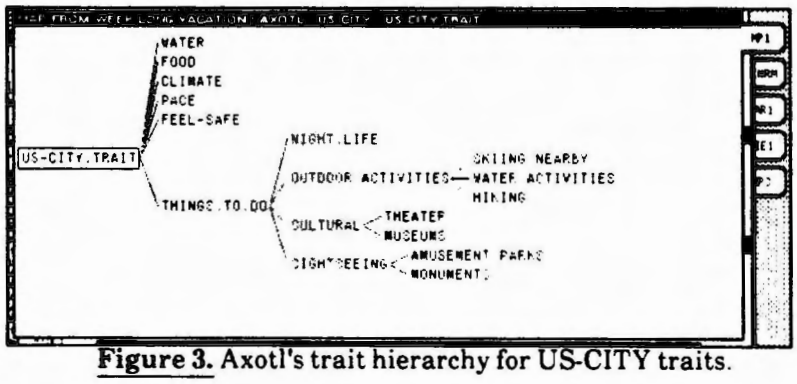



**Figure 3.** Axotl's trait hierarchy for US-CITY traits.

Nodes in the four hierarchies combine to form rating grids. In the most simple case, the children of a node in a solution hierarchy supply the solutions along the top of a grid; the children of a node in a trait hierarchy supply the traits down the side of a grid. Rating values within the grid are judgments about the solutions with respect to each trait.

Strategies for helping the expert build and refine hierarchies in Aquinas include laddering (eliciting specializations and generalizations through tailored dialogs), cluster analysis (experts are asked to label levels of abstraction represented by cluster junctions; see Figure 5), and trait value

examination (certain combinations of trait values may suggest that hiararchical expansion is appropriate).

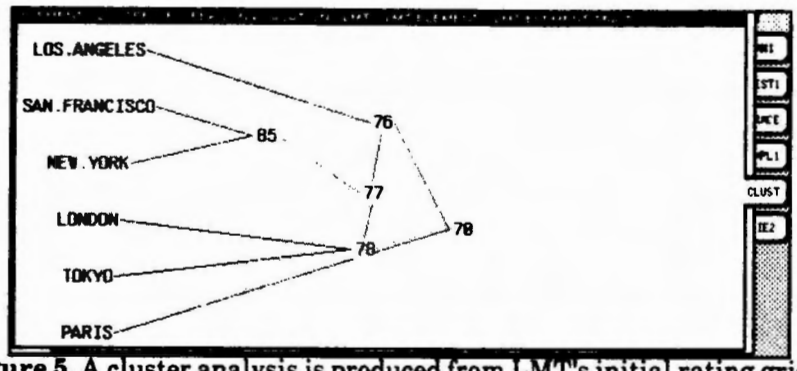



**Figure 5.** A cluster analysis is produced from LMT's initial rating grid. LMT is asked to form hierarchies by labeling junctions on the cluster. One likely distinction might be made between US and foreign cities.

**Rapid Prototypes of Consultation Systems.** The knowledge embedded in hierarchical rating grids in Aquinas can be used to solve specific problems by running *consultations*. During a consultation, Aquinas asks the user to specify the preferred or absolute values of traits associated with an ideal solution. Consultation results are displayed as rank-ordered, interval lists of solutions.

The model of problem-solving used in Aquinas is that of multiple knowledge sources (*experts*) that work together in a common problem solving context (*case*) by selecting the best alternatives for each of a sequential set of decisions (*solutions*). Alternatives at each step are selected by combining relevant information about preferences (relativistic reasoning), constraints (absolute reasoning) and evidence (probabilistic reasoning). This paradigm follows one suggested by Clancey who suggested that many problems are solved by abstracting data, heuristically mapping higher level problem descriptions onto solution models, and then refining these models until specific solutions are found (Clancey, 1986). A hybrid maximum entropy approach is applied by the reasoning mechanism.

Knowledge from multiple experts may be rapidly combined using Aquinas. Users may receive dissenting as well as consensus opinions from groups of experts, thus getting a full range of possible solutions. Disagreement between the consensus and the dissenting opinion can be measured to derive a degree of conflict for a particular consultation. The system can be used for cost-effective group data gathering (Boose, 1986b).

Weighted solution contributions from each expert are shown during consultations. When only one expert contributes to a solution, the weight is shown at 1.00 (for example, MIAMI). When more than one expert contyributes to a solution each expert's contribution weight is shown. In LOS.ANGELES for example AXOTL and LMT each contribute equally ( .5 ) so a weighted average of their contributions is computed as the total weight for the solution.

```
Test results for consultation MULTI-TEST for node WEEK.LONG.VACATION WEEK.LONG.VACATION.EXPERT
WEEK.LONG.VACATION.ELEMENT WEEK.LONG.VACATION.TRAIT

1 : MIAMI ( .74 )                    ( AXOTL 1.00 .74 )
2 : SEATTLE ( .72 )                  ( AXOTL 1.00 .72 )
3 : LOS.ANGELES ( .71 )              ( AXOTL .50 .78) (LMT .50 .64 )
4 : FRENCH.RIVIERA ( .70 )           ( AXOTL 1.00 .70 )
5 : SAN.FRANCISCO ( .63 )            ( LMT 1.00 .63 )
6 : AUSTIN ( .54 )                   ( AXOTL 1.00 .54 )
7 : NEW.YORK ( .52 )      ( AXOTL .50 .52) (LMT .50 .53 )
8 : PARIS ( .52 )                    ( AXOTL .50 .48) (LMT .50 .55 )
9 : COPENHAGEN ( .41 )               ( AXOTL 1.00 .41 )
10 : LONDON ( .28 )                  ( AXOTL .50 .18) (LMT .50 .39 )
11 : TOKYO ( .14 )                   ( LMT 1.00 .14 )
```

Next, a dissenting opinion is presented. Correlation scores for all experts are listed. A score of 1.0 would represent perfect ordering agreement (lists in the same order); a score of -1.0 would represent perfect disagreement (lists in opposite order). A score of 0.0 would mean that the lists are ordered randomly with respect to each other

Results from the expert with the "worst" score (the lowest) are presented side-by-side with the consensus opinion appearing above. Only those solutions that the dissenting expert knows about are listed in the comparative consensus. The order of consensus is preserved. In the results below, the only difference in ordering is that PARIS and NEW.YORK are reversed.

Users can get an idea of the overall agreement or disagreement among experts by comparing their correlation scores. Users can also see the entire range of acceptable solutions by comparing the consensus and dissenting opinions. In this case, even the most dissenting expert agreed that

LOS.ANGELES was a good choice and that TOKYO was not. The user gains more confidence that these are valid recommendations. If there are wide differences between experts, the user can use other tools in Aquinas to compare the appropriate rating grids to see where the experts' opinions differed.

```
Dissenting opinion for consultation MULTI-TEST for node WEEK.LONG.VACATION
WEEK.LONG.VACATION.EXPERT WEEK.LONG.VACATION.ELEMENT WEEK.LONG.VACATION.TRAIT

Correlation scores for all experts:
 LMT .84
 AXOTL .95

LMT has the most dissenting opinion ( .84 correlation score)

LMT / Consensus
1 : LOS.ANGELES .64 / LOS.ANGELES .71
2 : SAN.FRANCISCO .63 / SAN.FRANCISCO .63
3 : PARIS .55 / NEW.YORK .52
4 : NEW.YORK .53 / PARIS .52
5 : LONDON .39 / LONDON .28
6 : TOKYO .14 / TOKYO .14
```

**Automatically Generating Production Rules.** Aquinas can transform rating grids into sets of production rules. These rules can be automatically reformatted for several expert system building tools ( KEE, KS-300/EMYCIN, LOOPS, M.1, OPS5, S.1, and others; Boose, 1986). Certainty factors for rules are generated based on EMYCIN's certainty factor calculus (Adams, 1985). Certainty factor values are based on the relative strength of the rating, the relative weight of the trait, and the overall size of the grid.

**Process Guidance.** A subsystem of Aquinas called the dialog manager contains pragmatic heuristics to guide the expert through knowledge acquisition using Aquinas. Its help is important in the use of Aquinas, given the complexity of the Aquinas environment and the many elicitation and analysis methods available to the expert. The dialog manager makes decisions about general classes of actions and then recommends one or more specific actions providing comments and explanation if desired. This knowledge is contained in rules within the dialog manager in Aquinas. A session history is recorded so that temporal reasoning and learning may be performed (Kitto and Boose, 1987a). The dialog manager helps novices learn to use Aquinas, resulting in more efficient prototyping.

## Steps in Building a Rapid Prototype with Aquinas.

A typical sequence of steps when using Aquinas to build knowledge bases is:

1. Elicit cases and the initial grid (solutions, traits, and ratings)
2. Analyze and expand the initial, single grid
3. Test the knowledge in the single grid by running consultations
4. Build hierarchies (structured as solutions and traits in multiple grids) from the first grid if problem is complex
5. Use several rating value types (transform ordinal ratings to nominal and interval ratings) to represent knowledge
6. Test knowledge in hierarchies; add, analyze and test knowledge from other experts if warranted by application
7. If multiple experts are used, analyze the similarities and differences among experts; conduct structured negotiation among the experts (Boose, 1986b)
8. Edit, analyze, and refine the knowledge base, building new cases
9. Further expand and refine the knowledge base by successively testing the knowledge to see if the results agree with the expert's opinion

## Advantages and Disadvantages of Using Aquinas

Aquinas is useful for *analysis* problems whose solutions may be comfortably enumerated, such as classification, interpretation, and diagnosis. It is not as useful for *synthesis* problems where unique solutions are built up from component parts. Such problems include configuration, planning, scheduling, and design. However, sometimes Aquinas can be used on the analytic components of larger synthesis problems.

Aquinas can be used in a number of ways to help facilitate decision-making. Using Aquinas to help build a large, complex expert system is one such use. However, Aquinas can also be used as a stand-alone one-shot personal decision aid, a teaching aid, a group data gathering and negotiation tool, a tool for exploring project feasibility, and so on. Aquinas is useful in a variety of situations because of the flexibility and ease of use of rating grids and because of its ability to rapidly gather knowledge and build a prototype consultation system.

Aquinas inherits the advantages of ETS: rapid prototyping and feasibility analysis, vocabulary, solution and trait elicitation, interactive testing and refinement during knowledge acquisition, implication discovery, conflict point identification, expert system shell production, and

generation of expert enthusiasm (Boose, 1986a). It is much easier for users to learn knowledge-based system concepts by using Aquinas than through reading books or attending classes (i.e., rules are automatically generated and used dynamically in consultations; new vocabulary is incrementally introduced).

ETS, the predecessor of Aquinas, only represents knowledge in a single rating grid. It has been employed to build hundreds of single-grid prototype systems, and is still in use at Boeing. Single grids as large as 42 x 38 (1596 ratings) have been built. Knowledge bases have been generated for expert system shells that contain over two-thousand rules. Typically, something on the order of a 15 x 10 grid is built with ETS that generates several hundred rules.

Over 50 prototype systems have been built during the development and introduction of Aquinas (released in Boeing in April, 1987). Some of these systems contain thousands of ratings arranged in hierarchical grids.

Knowledge from multiple experts may be rapidly combined using Aquinas. Users may receive dissenting as well as consensus opinions from groups of experts, thus getting a full range of possible solutions. Disagreement between the consensus and the dissenting opinion can be measured to derive a degree of conflict for a particular consultation. The system can be used for cost-effective group data gathering (Boose, 1986b).

Elicitation, structuring, analysis, and testing of knowledge is based on specific cases. When knowledge in Aquinas is updated, it is done so with respect to a specific case. Addition of new knowledge in this way can be strictly controlled by the expert; the tendency for local changes to degrade other cases is thus curbed.

## Future Directions and other Knowledge Acquisition Tools

We intend to build a knowledge acquisition environment that includes specific domain knowledge for specialized application areas and can acquire knowledge for synthetic problems, combining features from other knowledge acquisition tools. Development of the Aquinas workbench will continue in an incremental fashion. Techniques will be continuously integrated and refined to build an increasingly more effective rapid knowledge acquisition environment.

## Acknowledgments

Thanks to Roger Beeman, Miroslav Benda, Kathleen Bradshaw, William Clancey, Brian Gaines, Cathy Kitto, Ted Kitzmiller, Art Nagai, Doug Schuler, Mildred Shaw, David Shema, Lisle Tinglof-Boose, and Bruce Wilson for their contributions and support. Aquinas was developed at the Knowledge Systems Laboratory, Advanced Technology Center, Boeing Computer Services in Seattle, Washington.

## References

Adams, J., Probabilistic Reasoning and Certainty Factors, in B. Buchanan and E. Shortliffe (eds.), Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, MA: Addison-Wesley, 1985.

Brown, D. E., Expert Systems for Design Problem-Solving Using Design Refinement with Plan Selection and Redesign, unpublished Ph.D. dissertation, Ohio State University, CIS Department, Columbus, Ohio, Aug., 1984.

Buchanan, B., and Shortliffe, E. (eds.), Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, MA: Addison-Wesley, 1985.

Benda, M., Baum, L. S., Dodhiawala, R. T., and Jagannathan, V., Boeing Blackboard System, in the *Proceedings of the High-Level Tools Workshop*, Ohio State University, October, 1986.

Boose, J. H., Personal Construct Theory and the Transfer of Human Expertise, in the *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, 1984.

Boose, J H., A Knowledge Acquisition Program for Expert Systems Based On Personal Construct Psychology, *International Journal of Man-Machine Studies*, 23, 1985.

Boose, J H., Expertise Transfer for Expert System Design, New York: Elsevier, 1986a.

Boose, J. H., Rapid Acquisition and Combination of Knowledge from Multiple Experts in the Same Domain, *Future Computing Systems Journal*, Vol 1., No. 2, 1986b.

Boose, J. H., and Bradshaw, J. M., Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge Acquisition Workbench for Expert Systems, special issue on the AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, *International Journal of Man-Machine Studies*, in press, 1987.

Clancey, W., Heuristic Classification, in J. Kowalik (ed.), Knowledge-Based Problem-Solving, New York: Prentice-Hall, 1986.

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

Gaines, B. R., and Shaw, M. L. G., New Directions in the Analysis and Interactive Elicitation of Personal Construct Systems, in M. L. G. Shaw (ed.), Recent Advances in Personal Construct Technology, New York: Academic Press, 1981.

Gale, W. A., Knowledge-Based Knowledge Acquistion for a Statistical Consulting System, to appear in a special issue on the AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, *International Journal of Man-Machine Studies*, in press, 1987.

Kelly, G. A., The Psychology of Personal Constructs, New York: Norton. 1955.

Kitto, C. M., and Boose, J. H., Heuristics for Expertise Transfer: The Automatic Management of Complex Knowledge Acquisition Dialogs, special issue on the AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, *International Journal of Man-Machine Studies*, in press, 1987a.

Kitto, C. M., and Boose, J. H., Choosing Knowledge Acquisition Strategies for Application Tasks, the Proceedings of Westex-87, Anaheim, 1987b.

Shaw, M. L. G., and Gaines, B. R., Techniques for Knowledge Acquisition and Transfer, special issue on the AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, *International Journal of Man-Machine Studies*, in press, 1987.

paper for Knowledge Acquisition Workshop (Reading, GB)

# Knowledge Acquisition as modeling expertise: the KADS methodology

*Joost Breuker*

*Bob Wielinga*

Department of Social Science Informatics
University of Amsterdam
Herengracht 196
1016 BS Amsterdam, The Netherlands

email: mcvax!swivax!breuker (or: breuker@swivax.UUCP)

## 1. KADS Methodology

Although there is ample recognition of the fact that 'Knowledge Acquisition' is the major bottleneck in developing expert systems thus far little guidance (methodology) or support tools have been developed to overcome this bottleneck. This is in a sharp contrast with the abundance of tools (e.g. 'shells) which support the implementation of expert systems.

Knowledge acquisition consists of the collection, elicitation and interpretation of data on the functioning of expertise in some domain, in order to design, build, extend, adapt or modify a knowledge based (expert) system. In this view, knowledge acquisition is a permanent and crucial activity throughout all stages of designing, implementing and maintaining an expert system.

Knowledge acquisition for Knowledge-based systems is a difficult task, which is generally practised as an art rather than as a well-structured engineering discipline. The commercial application of Knowledge-based systems requires a more structured and better documented approach than is currently available. Breuker and Wielinga (1984; 1987) have proposed a framework for a methodology for Knowledge Acquisition based on the presupposition that analysis of data obtained from human experts and written sources can be decoupled from the design and implementation stages of the process of Knowledge-based systems construction. This methodology -the KADS (Knowledge Acquisition and Structuring) methodology- aims at a more structured approach to knowledge-based system development than the currently practised experimental approach. In addition, the methodology is supported by a set of tools that support the activities of the knowledge engineer. This set of tools is integrated into the KADS system. It is a *Knowledge Engineers Workbench.*

The KADS methodology/system supports both a bottom-up and a top-down approach to knowledge acquisition. In this paper we will concentrate on the top-down approach, i.e. describe how models can be used to guide the knowledge engineer in analysing the data on expertise in a particular domain. Before discussing modeling expertise we

The research reported here is partly funded by the ESPRIT Programme of the European Community under contract P1098, 'Methodology for the development of knowledge based systems'. Partners in this project are: STC, Scicon, KBSC of the Polytechnic of the Southbank (GB), SCS (Westgermany), Cap Sogetti (France) and the University of Amsterdam.

will briefly describe the KADS methodology as it is currently practised.

The KADS methodology is based upon a number of principles, which are derived from methodologies in various disciplines (psychology, AI, software engineering) and experiences in building expert systems. The principles can be viewed as epitomizing the support knowledge of KADS, and can be read as high level guidelines in knowledge analysis for expert systems. The principles can be stated as follows:

* The knowledge and expertise should be analysed before design and implementation starts; i.e. the major efforts in knowledge acquisition should occur before an implementation formalism is chosen.
* Because of its inherent complexity, the analysis should be model driven as early as possible. Models of expert problem solving not only enable -at least, facilitate- the analysis of data, but also provide references to known implementation solutions (cf. ROGET; Bennet, 1985).
* To sail between the Scylla of natural language (with ambiguous but rich semantics) and the Charibdis of a formal language (clean but unexpressive semantics), a model of expert problem solving should be expressed at the epistemological level (eg. Clancey, 1985).
* The analysis should not only include the knowledge and reasoning of the expert(s), but also the functionality of the prospective system, i.e. data on the environment and users should be collected and analysed. These data are used for defining the communication tasks -modality- of the system.
* The analysis should proceed, preferably, in an incremental way. Because there is a wide variety of topics, such as tasks, functions, knowledge, etc., which are not unrelated, the analysis should be breadth-first and cyclic to allow successive refinement and inclusion of results of analysis of related topics.
* New data should only be elicited when the analysis of previously collected data shows the need for clarification and augmentation. As a result, elicitation and analysis will alternate. This should both keep the data manageable, and lead to more specific topics for -and methods of- elicitation (Breuker and Wielinga, 1984).
* Collected data and interpretations should be documented. On line documentation of the analysis is a conditio sine qua non for team work. The 'Domain Handbook' (Grover, 1983) is also an excellent guide for system maintenance.

Not all of these principles and premisses will be worked out in this paper. (c.f. Breuker & Wielinga, 1985; 1987; Wielinga & Breuker, 1984, 1986).

We will focus on the role of models, and in particular the role of interpretation models. This means that we will work out the first three premises above, in particular the top-down support of interpretation models and the bridging function of models which are formulated at the epistemological level.

## 2. Role of Models

### 2.1. Conceptual Models of Expertise

A knowledge based system is a model of some expertise. Like all modelling activities, it involves the transformation of a set of data into some abstract structure. A model 'reveals' an implicit structure in a set of data, and in that sense explains or interprets the data. By abstracting from some point of view, a model focusses on a subset of all possible relationships between data, or elements that correspond to some subset of

data. Modelling is a complicated form of design. In design a structure is generated that satisfies constraints (requirements). In modelling the constraints are more severe in the sense that some form of correspondence should be maintained between the data and the structure. Which correspondence is dependent on the purpose of the model.

The purpose, or main function, of a knowledge based system is not to explain, or interpret data, but to solve problems. This means that the data are descriptions of problems and solutions; the model provides the mapping between these. This is probably a too pragmatic, behaviouristic view on modelling expertise. It would also apply to most conventional systems. However, the problems that have to solved by a knowledge based system are of a different level of complexity than those solved by conventional systems. It is this complexity that makes this mapping less straightforward, and requires additional data: about knowledge and its use.

The problems to be solved by a KBS are not decomposable in some obvious way, i.e. without the required expertise. In fact, many expert tasks consist mainly of a problem analysis that should lead to the identification of a known solution, as e.g. in diagnosis. In design and planning tasks ("system synthesis", Clancey, 1985) the solution has to be constructed as well, which makes these types of tasks considerably more complicated for KBS development -and also: exclusively fit for KBS rather than conventional systems.

In expert tasks the decomposability (analysis) of a particular problem -or class of problems-, may be spelled out, and implemented as a fixed route, but this route is only applicable to the same, or a small subset of typical domain problems. Capturing the large variety in which problems in a domain of expertise may manifest themselves requires abstractions that distinguish the 'themes' and 'variations' in these problems. This is what the knowledge base and its inference competence is for; the way to decompose the problem (or to construct solutions) is the problem of how to control the use of this competence.

A consequence of the relative complexity and variability of problems that have to be solved by some KBS or expert is that some types of problems are solved in a different way than other problems. This means that the use of knowledge should be modelled in such a way that the knowledge base can be exploited in a *flexible* way. Flexibility not only implies different approaches for different types of problems, but more importantly, the ability to distinguish solvable from non solvable problems ('graceful degradation'). This is of practical importance: false solutions are often more costly than no solutions, in particular when false solutions can only be recognised by their 'real world effects', i.e. after the fact.

Current implementation formalisms do not support high level modelling of flexible expertise, as found in human expertise. In the form of shells, the formalisms are too rigid, and only tuned to a particular type of task. As environments the implementation formalisms provide too low level elements. The fact that implementation languages are at too low a level for specification of a system has led to the use of special specification languages in conventional software design, and such specifications are the output of a domain analysis. In the KADS methodology we are aiming for a similar instrument. The specification languages for conventional software development cannot be applied, because they assume a way of decomposing the problem in which at each level the same elements can be applied. However, as we will show in the next chapter, it seems more adequate to decompose expertise in functionally differentiated layers, i.e. for each layer different elements are used.

To summarise: developing a KBS involves modelling of expertise, i.e. the structuring of the required knowledge, inference competence and (flexible) use of knowledge. The means to directly express this is not provided by current implementation formalisms, similarly to what is the case in conventional software engineering. Likewise, in developing a KBS an Analysis stage should precede the implementation, and this Analysis stage should produce (i.a.) a high level model (specification) of the expertise required for building the system. This model is called **conceptual model** (Norman, 1983). It should be noted that the conceptual model requires further transformations and specifications to obtain a *design model*, which is too some extent a functional equivalent of the conceptual model, but now in terms of implementation formalisms, or rather: AI techniques, which lend themselves for coherent implementation. The role of a conceptual model is therefore similar to a functional specification, and does not only provide a coherent view on expertise in a particular domain, but also supports making explicit decisions on the design and implementation of (flexible) knowledge based systems.

### 2.2. What's in a model of expertise

In order to obtain the required flexibility, we have proposed a four layered structure for models of expertise. For more extensive descriptions of this four layer structure see Wielinga & Breuker (1986; also: Breuker et al., 1987). Here we present a short overview of the successive layers and their roles.

The first layer is called the *domain layer*. In this layer knowledge is represented as concepts, relations between concepts, and structures build from relations. The concepts are domain dependent. Relations in a domain represent facts, empirical associations, structural, functional, and attributive relations, etc. Knowledge at the domain level permits some problem solving processes, but does not contain facilities to control such problem solving. This layer is task neutral, which means that -in principle- the knowledge specified at this level may be used for a variety of (sub)tasks, and therefore, can be flexibly applied.

The next layer provides at a meta level the *inference competence* that is implied by these knowledge structures (and inference methods). The inference structure describes which inferences can be made, but not how these inference making functions are controlled at the task level. The roles that the domain level structures -and their interpretation- play are specified at the inference level in terms of *knowledge sources* and *metaclasses*. Metaclasses describe the role of a concept in the problem solving process. For instance, a hypothesis is such a role, and in a medical diagnostic domain this may stand for an infection. The knowledge sources are the inference making functions, which have metaclasses as input/outpout arguments. Knowledge sources indicate which type of inference is required to proceed from an input- to an output metaclass. Metaclasses and knowledge sources have dependencies which form the *inference structure*. An example of an inference structure is presented in figure 1.

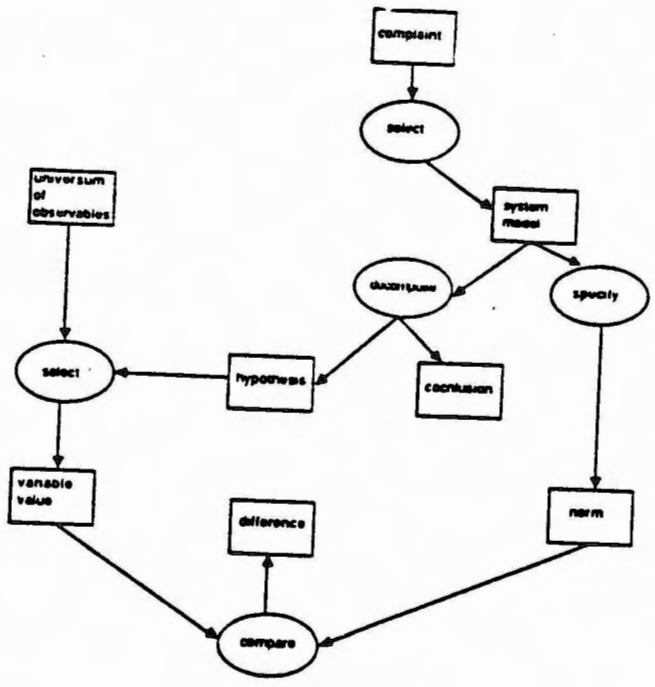



Figure 1. Inference structure of systematic diagnosis.

At the *task* level the goal/task decomposition of the problem solving process is described. The task structure specifies control over the inference structure. The goals are either generated at the next, strategy, level, or by tasks themselves.

The fourth layer is the *strategy* layer. It represents the strategic reasoning about the problem solving process. At this (metalevel) one can assume a full problem solver which is capable to plan the problem solving procedures at the domain level by using elements of the inference structure as 'handles'. Moreover, at this level the execution of the problem solving is monitored, and if an impasse occurs, a diagnosis may lead to remedials (cf. 'repair theory', Brown & de Kleer, 1979), or to a replanning of the problem solving procedures.

### 2.3. Interpretation Models

Although expertise may come in many varieties, and varieties within these varieties by ranges of problems, we hold the assumption that expertise can be classified according to the global structure of the reasoning process. Intuitively, trouble shooting of physical devices is more akin to diagnosing biological systems, than designing a physical system. In diagnosis and in trouble shooting hypotheses are generated and tested about malfunctions, while in a design task elements are configured into a solution that satisfies many constraints. Domains may contain different concepts, but tasks may share ways of using knowledge at some higher level of abstraction. The reason they may share use of knowledge is by the fact that not only the structure of their problems and solutions may be similar, but also by the fact that the structures of knowledge that support this reasoning process may be similar. A tentative classification of elementary, or 'generic' tasks is presented in Figure 2.

```
system_analysis
|   identify
|   |   classify
|   |   |   simple_classify
|   |   |   diagnosis
|   |   |   |   single_fault_diagnosis
|   |   |   |   |   heuristic_classification
|   |   |   |   |   systematic_diagnosis
|   |   |   |   |   |   causal_tracing
|   |   |   |   |   |   localisation
|   |   |   |   multiple_fault_diagnosis
|   |   |   assessment
|   |   monitor
|   predict
|   |   prediction_of_behaviour
|   |   prediction_of_values
system_modification
|   repair
|   remedy
|   control
|   |   maintain
system_synthesis
|   transformation
|   design
|   |   transformational_design
|   |   refinement_design
|   |   |   single_stream_refinement_design
|   |   |   multiple_stream_refinement_design
|   |   configuration
|   planning
|   modelling
```

Figure 2: Taxonomy of generic problem solving tasks

Evidence for similarities at the level of inference competence and control of use of knowledge comes both from psychometric studies (e.g. Guilford, 1967, Sternberg, 1982), modelling of human problem solving (e.g. Elshout, 1976), and knowledge engineering (e.g. Clancey, 1985; Chanderasekaran, 1983; Hayes-Roth et al., 1983). This means that conceptual models of expertise in different domains -but representing similar tasks- may have the same higher level structure. Thus a conceptual model can be abstracted from its domain specific content and function as a skeletal model for the conceptual model of expertise in another domain, but with a similar task. Such an abstracted model is called **interpretation model**. The role of an interpretation model is a template structure that allows for (some) top down refinement instead of bottom up construction of the conceptual model in the analysis of expertise in a particular domain. An interpretation model is an intermediary between "what to look for in the data" and the conceptual model that is to be constructed. In the KADS methodology, a conceptual model is a product; an interpretation model is a tool, that is used 'whenever appropriate' during the analysis. An example of the inference structure of an interpretation model is presented in figure 1. The inference structure is the most typical and unique structure of an interpretation model. Within a generic task there may be varieties of structures; not only at the domain level, but also the taskstructures may vary to some extent, dependent on characteristics of the domain.

## 3. Current state

We have developed interpretation models for most of the generic tasks presented in figure 2 (see Breuker et al., 1987), which constitute a library of models in the KADS system (see Anjewierden, 1987). Although the theoretical basis for these interpretation models appears to be far more complicated than the simple formalisms currently used in developing expert systems, experiences, both within Esprit project 1098 and in commercial practice, have shown that the use of these models strongly facilitates the acquisition of a coherent view on the domain. Lack of such a coherent, top down view is exactly one of the reasons current practice in developing expert systems is characterised by inefficiency (backtracking, throwing away prototypes) and intransparency (teamwork is impossible; progress can hardly be monitored).

## 4. References

Bennett, J. (1985) ROGET: a knowledge-based system for acquiring the conceptual structure of a diagnostic expert system, *Journal of Automated Reasoning*, vol.1, 49-74.

Brachman, R.J. (1979) On the Epistemological status of semantic networks. (1978) In: N.V. Findler (Ed.) *Associative Networks* New York: Academic Press.

Breuker, J. A., and Wielinga, B. J. (1985) KADS: structured knowledge acquisition for expert systems. *Proceedings of Fifth international workshop on expert systems and their applications*. Avignon, France.

Breuker, J.A. (1987) (ed) Model driven knowledge acquisition: interpretation models, Deliverable A1, Esprit 1098, University of Amsterdam

Brown, J.S. & DeKleer, J., (1979) Repair Theory, *Cognitive Science*

Chandrasekaran, B. (1982) Towards a taxonomy for problem solving *AI Magazine*

Clancey, William J., (1983) The Epistemology of a Rule-Based Expert System - a Framework for Explanation, *Artificial Intelligence*, 20, 215-251.

Clancey, W. J. (1985) Heuristic classification. *Artificial Intelligence*, *27*, 215-251.

Clancey, W.J. (1985b) Acquiring, representing and evaluating a comptence model of diagnostic strategy. To appear in: Chi, Glaser & Farr, *Contributions to the nature of expertise*

Elshout, J.J. (1976) *Karakteristieke moeilijkheden in het denken*, University of Amsterdam.

Guilford, J. P. *The nature of human intelligence*. New York: Academic Press, 1967.

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

Hayward, S.A., Wielinga, B.J. and Breuker, J.A. (1986) Structured analysis of knowledge, *Proceedings of Knowledge Acquisition for KBS workshop, Banff, Canada.*

Norman, D.A. (1983) Some observations on mental models. In D. Gentner & A. Stevens (eds) *Mental Models*, Hillsdale, Erlbaum

Wielinga, B.J. & Breuker, J.A. (1984) Interpretation models for knowledge acquisition. T. O'Shea (ed): *Proceedings of Advances in Artificial Intelligence*, (ECAI, Pisa), Amsterdam: North-Holland.

Wielinga, B.J. and Breuker, J.A. (1986) Models of expertise, *Proceedings of ECAI 1986*.

# HOW DO EXPERTS ACQUIRE EXPERTISE?

Brian R Gaines
Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4

**Abstract**: The role of the expert in the knowledge processes of society is analyzed and shown to be that of managing the inductive dynamics of knowledge acquisition. Experts trade the skills based on their knowledge for access to problems providing them with new material for the inductive process. The formation of expertise is functional in general because it leads to division of labor in the management of knowledge acquisition. The development of an individual expert is a random process brought about by strong positive feedback loops in the social process, for example, that a proto-expert with superior performance is brought more problems and hence has a greater opportunity to learn and improve that performance. A diversity of such positive feedback processes operate in the professions and sciences with little relation between them except their overall effect in promoting the formation of expertise. Sociological evidence for the existence of these effects is discussed, and simulation studies showing the quantitative phenomena are described.

## INTRODUCTION

Knowledge acquisition and transfer processes in human society are complex and poorly understood (Gaines 1987a). Expert systems add a new component to these processes by enhancing an expert's capability to transfer knowledge in a usable form. However, taking part of the social process of knowledge transfer and embedding it in technology makes it necessary to have a more overt understanding of what an expert is, and does. It also raises questions about the possibly changing role of the expert—what will the expert do once the expertise is captured—who is accountable for the advice given by the expert system?

This paper is the second in a series addressing the problems of developing knowledge support environments for the development, application and validation of knowledge-based systems. The first paper (Gaines 1987a) analyzes the foundations of knowledge-based systems. A distributed anticipatory system formulation of knowledge acquisition and transfer processes is presented which provides scientific foundations for knowledge engineering. This paper builds on this formulation to present a model of experts as managers of knowledge acquisition and transfer processes in particular domains, trading the transfer of their existing knowledge for the access to new problems that will enable them to acquire more. The third paper (Gaines 1987b) develops an architecture for knowledge-based systems, integrating knowledge acquisition, application and validation.

## THE NATURE OF EXPERTISE

Hawkins (1983) has abstracted from experience at Schlumberger in developing mineral exploration expert systems and proposed a model of human expertise relevant to expert systems. His model is summarized in figure 1:

- The expert first elicits data about the problem from the client;
- He develops a minimal model that accounts for the data provided;
- He generates advice based on the model and feeds this back to the client;
- The client may accept the advice, or query it and, possibly, the model;
- The queries lead to further data elicitation, and repeat of the modeling/advice/query cycle.

Thus, in Hawkins' model, the client plays an active role in further developing the model by providing more data until he or she is satisfied with the model and consequent advice. Thus, expert advice giving and taking is part of a cycle of negotiation around a process of model formation.

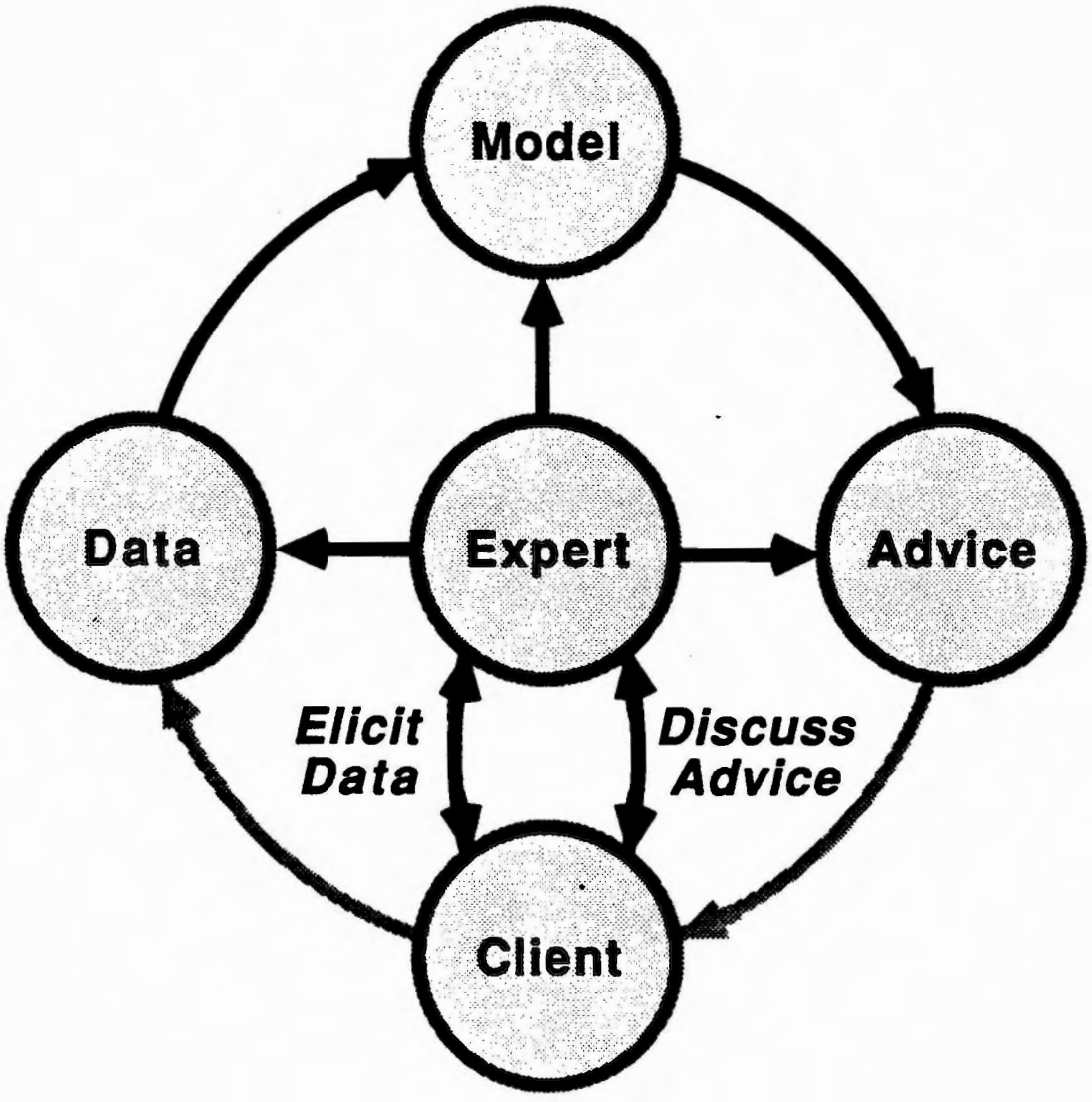



**Fig.1 The negotiation cycle in expert consultation**

This model of the expert-client interaction as the negotiation of a mutually acceptable model may be given formal foundations in mathematical modeling theory. Figure 2 shows a general hierarchical modeling process (Gaines 1987a) based on Klir's analysis of model formation (Klir 1985):

- At level one, *constructs* are those distinctions that the particular modeling system makes, a language for describing of the world;
- At level two, *data* are descriptions of actual case histories in terms of the constructs, an account of experiences of the world;
- At level three, *hypotheses* are the means of regenerating particular case histories from generalized accounts, rationalizations of the world (often called *models*);
- At level four, *analogies* are similarities between differently generated generalized accounts, correspondences between models;
- At level five, *abstractions* are accounts of a wide range of models, underlying foundations of analogies;
- At level six, *principles* are systemic foundations for abstraction, accounts of abstractions.

Rasmussen (1985) has noted the importance, and analyzed the roles, of such hierarchical knowledge representation in individual and organizational decision making.

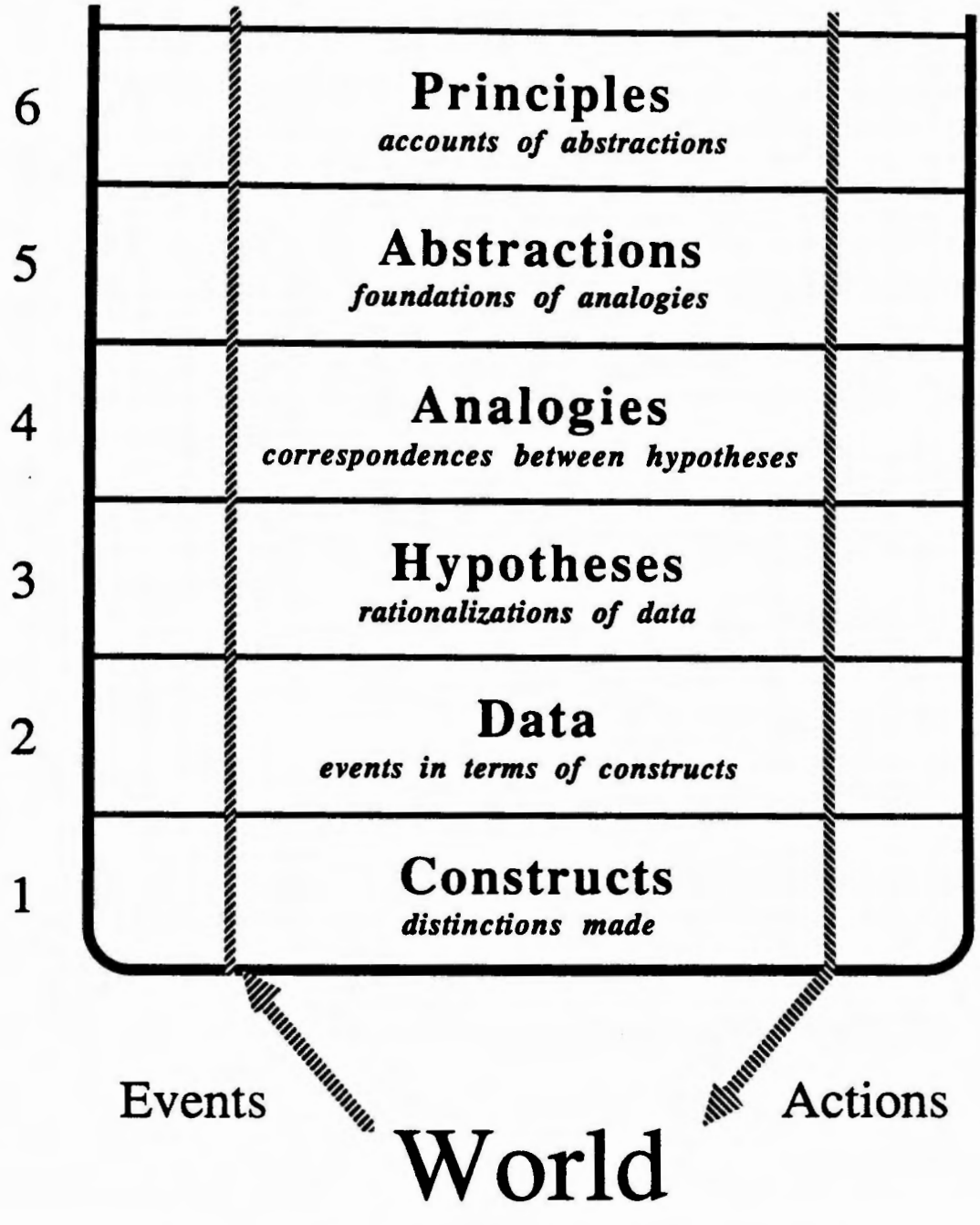



**Fig.2 General hierarchical modeling process**

Modeling may be evaluated by the degree to which a level accounts for the information flowing through it. This may be measured as the *surprise* at an actual event in relation to one predicted, and the expected surprise is a form of entropy (Gaines 1977). The formal theory of modeling is one in which selections are made at each level down the hierarchy to minimize the rate at which surprise is passing up the hierarchy. The criteria for selection independent of the data are generally thought of as being ones of simplicity/complexity: of two models which fit the data equally well choose the simplest. The simplicity/complexity ordering is arbitrary and in its most general form is just one of *preference*. Hence the general modeling schema is one in which surprise flows up the hierarchy and preference flows down. It should be noted that the

modeling is an active process also—constructs are not just static partitions of experience. They may be operations: actions in psychological terms; processes in computational terms. Whether a system finds a distinction in the world, imposes it passively as a view of the world, or imposes it actively as a change in the world, is irrelevant to the basic modeling theory. Figure 2 encompasses both the expert's model-formation and model-application processes.

Hawkins' (1983) studies exemplify the roles of each of these levels in expert behavior and in the negotiation between expert and seeker of advice. In particular, he illustrates the significance of shared constructs between expert and user, and the potential for 'data' from falsely constructed events to invalidate expertise. What the basic modeling theory does not account for, however, is these types of social interaction, the relation between experts and users, the assumptions that an expert makes about the problem situation, the relation between experts and society. Indeed, it does not account for the concept of an 'expert' except as a particularly well-validated modeling system, leaving open the question as to how one system becomes better validated than another.

The following section extends the basic modeling theory to encompass social processes and the formation of expertise.

## SOCIAL PROCESSES IN THE FORMATION OF EXPERTISE

The weakness of the account of modeling given in Figure 2 is that it accounts for only a minor part of the knowledge acquisition processes of an individual. Most of our knowledge does not come from forming models of events in the world, but rather from the social transfer of others' knowledge by a variety of cultural processes, some formalized in education but many very informal. However, the account can be extended to encompass such cultural knowledge acquisition if one notes that human civilization as a whole is representable as a general modeling system having no sources of knowledge other than its combined, cumulative experience of the world (Gaines 1987a).

The modeling schema of Figure 2 applies as much to groups of people, companies and societies as it does to the roles of a person. The modeling processes of an individual may be viewed as a cross-section of those of the civilization of which they are part. The modeling processes of society may be viewed as carried out by a distributed system of sub-processes in individuals. Cultural processes of knowledge transfer are then just various modes of communication between such sub-processes. The distributed system analogy nicely accounts for a variety of psychological and social phenomena: one person may assume many psychological roles (process switching), whereas a group of people working together may act as a single modeling entity and hence behave as one process (distributed processing).

These concepts may be formalized by considering the inductive inference process underlying knowledge acquisition. Deduction guarantees to take us from valid data to valid inferences, but the inferences are thereby part of the data—no new knowledge is generated. Induction takes us from valid data to models of that data that go beyond it—by predicting data we have not yet observed, and by giving explanations of the data in terms of concepts that are unobservable. Induction generates new knowledge but, as Hume (1739) pointed out over 200 years ago, the process is not deductively valid and it is a circular argument to claim that it is inductively valid. Goodman (1973) proposed that we accept the circularity but note that it involves a dynamic equilibrium between data and inference rules as shown in Figure 3:

> "A rule is amended if it yields an inference we are unwilling to accept; an inference is rejected if it violates a rule we are unwilling to amend."

Rawls (1971) in his theory of formal justice as induction from informal moral behavior terms this a *reflective equilibrium*.

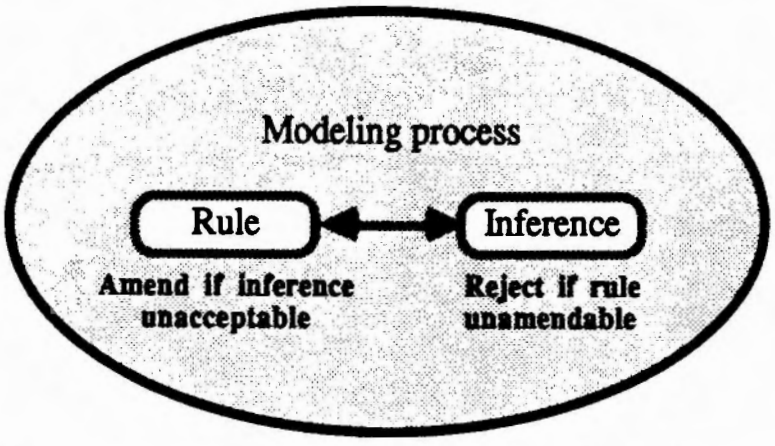



**Fig.3 Reflective equilibrium in inductive inference**

Stich and Nisbett (1984) noted flaws in Goodman's argument in that individuals do not necessarily behave rationally in their own behavior relative to known good practice. They repaired them by proposing that the equilibrium is social not individual:

"a rule of inference is justified if it captures the reflective practice not of the person using it but of the appropriate experts in our society."

This gives an operational definition of the role of experts within an overall knowledge acquisition system as referential sub-systems managing a domain of the knowledge acquisition process. It also leads to an concept of the role of expert systems in society, as making the referential process more overt and widely available. It is noteworthy that professional development books concerned with good practice emphasize the importance of balancing experts' authoritarian roles as referential sub-systems with their entrepreneurial roles as knowledge acquirers (Schön 1983).

The hierarchical model of Figure 2 can also be used to give a theoretical framework for the socio-cultural transmission of knowledge. The left hand side of Figure 4 shows the feedback loops between events and actions at each level of the hierarchy:

- At level one, *reflexive* feedback links the activation of a construct directly to a primitive act, another construct;
- At level two, *rule-based* feedback links constellations of experience to complete action sequences through rules derived from similar experience;
- At level three, *computational* feedback uses a generative model of experience to compute an optimal action sequence;
- At level four, *comparative* feedback compares a variety of alternative models in order to one appropriate to the required goals;
- At level five, *abstract* feedback uses generalized models as templets from which to instantiate one appropriate to the required goals;
- At level six, *transcendental* feedback takes the entire process described, recognizes that it is based on distinctions being made at various level, and attempts to rationalize these distinctions and create new ones.

The acquisition of knowledge by direct inductive inference, learning a task by attempting it, is the only course open to the species as a whole. However, it would be inefficient for individuals to acquire more than a small part of their knowledge in this way, and the right hand side of Figure 4 shows the cultural mechanisms through which individuals acquire knowledge at each level:

- At level one, mimicry through watching expert's perform the task is the main mechanism for transmitting reflexive knowledge.
- At level two, reinforcement through working under expert supervision is the main mechanism for transmitting rule-based knowledge.
- At level three, rational explanation by reading books and interviewing experts is the main mechanism for transmitting computational knowledge;

- At level four, analogical models used to derive knowledge from similar situations form the main mechanism for transmitting comparative knowledge;
- At level five, basic laws used to derive specific models are the main mechanism for transmitting abstract knowledge;
- At level six, systemic principles used to derive specific laws are the main mechanism for transmitting transcendental knowledge.

This plethora of mechanisms for knowledge transmission is further complicated by the possibilities of *managing* the learning situation, for example, one person may route another to a good role model, supervisor, explainer, similar task, or underlying law or principle.

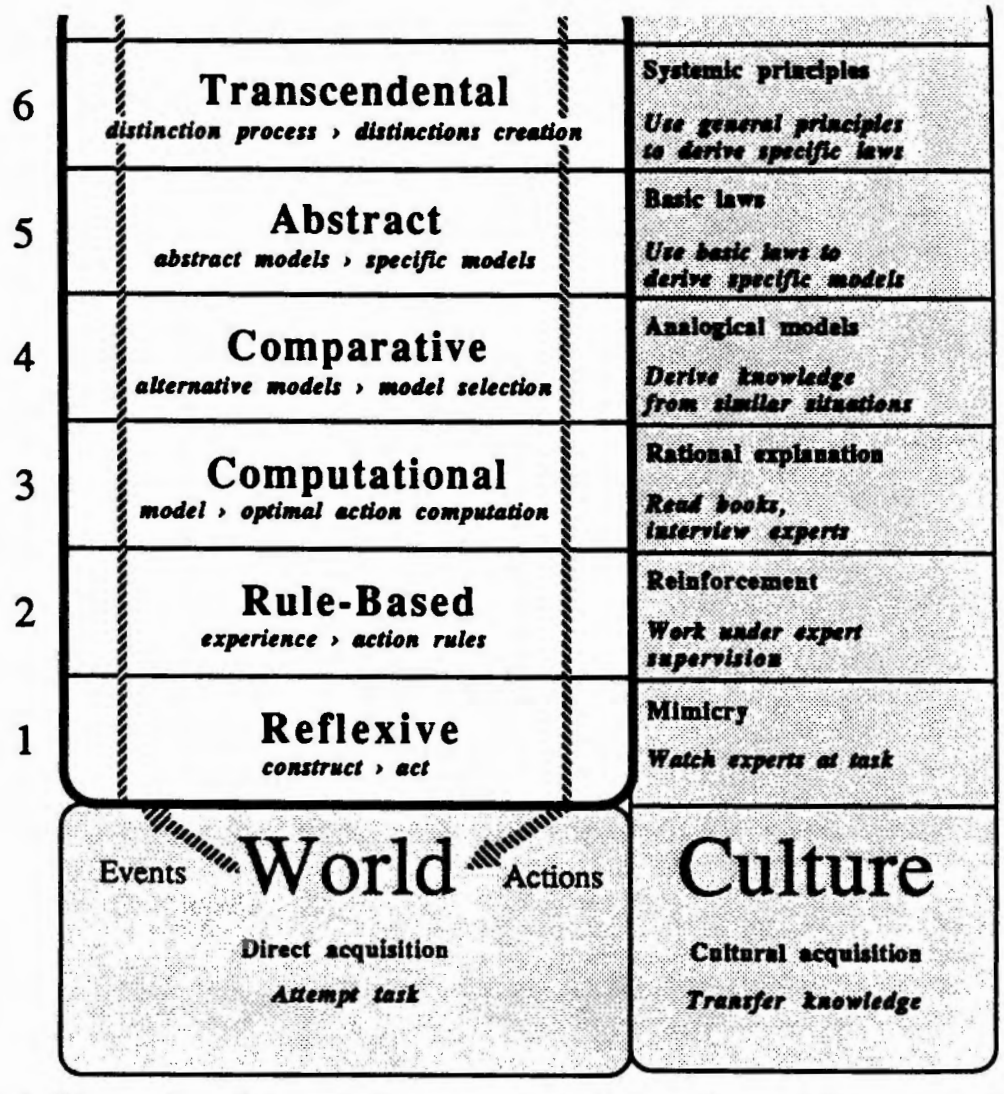



**Fig.4 Hierarchy of knowledge representation and acquisition processes**

Thus the pragmatic model of expertise developed by Hawkins can be brought into the formal framework of modeling theory by taking society to be a distributed modeling system in which experts are those with responsibility for managing the inductive process in particular domains. However, even this extension does not specifically account for the formation of expertise—how does a particular individual become an expert. The following section shows that positive feedback processes in the management of the various levels of knowledge transmission driven by random processes in society is adequate to account for expertise formation.

## EXPERTISE FORMATION BY POSITIVE FEEDBACK PROCESSES

An expert in Stich and Nisbett's account is a modeling system that has particular authority to decide on changes when events violate rules—in terms of Figure 2, are new constructs needed, should the data be rejected, should the hypothesis be changed, and so on. However, this authority is a social construct, not usually one of right in present society, and the ascription of such authority to an individual is part of the larger modeling system's processes in determining which of its sub-processes is effective. Now, to become effective in modeling is, from Figure 4, dependent on many factors: having access to knowledge at the upper levels such as principles, abstractions, analogies and classes of hypothesis; having access to knowledge at the lower level such as adequate construct systems and relevant data from case histories. Such access is itself a social resource, not generally available to all, and there are social mechanisms for allocating such access to individuals. It is the interaction between the criteria for access and the acquisition of knowledge that leads to the formation of expertise.

Sociologists have noted the very strong positive feedback processes in the dynamics of the scientific community (Hagstrom 1965). Merton (1973) coined the term the "Matthew effect" for those features of the reward system in research that were biased towards allocating greater credit for the same discovery to those with an already established reputation. The qualitative effect of such positive feedback processes is to amplify random differences within a population, creating strong distinctions which have no other basis than the feedback process itself. That is, even if all scientists were created equal in their capabilities, some would become experts relative to others because random differences in early performance generated credit that affected resource allocation and gave them greater opportunities to acquire knowledge.

In many areas of expertise where expert systems are expected to play a major role these positive feedback processes are exceptionally strong. In medicine, for example, the key learning resource is access to medical problems, and the 'owner' of such a problem has a keen personal interest in only allowing someone of very good reputation to handle it. The system, including considerations of legal liability, strongly funnels problems to those who are regarded as experts. It is, however, precisely these problems from which new knowledge is generated. Hence the data at level two in Figure 2 is made available primarily to those with already established reputations for having capabilities at level three and above to deal with it. Similar considerations apply to the award of scholarships, invitations to scientific congresses, and so on (Blume 1974). There are many positive feedback loops that seek out expertise and seek to develop it. However, they are also adequate in their own right to create it.

Simple simulation experiments are adequate to show the strength of these phenomena. Figure 5 shows the structure of a simulation in which a user community is represented as an *expert selection automaton* which determines which of two experts, represented by *expert automata*, it will use in the solution of a succession of problems. The rules are: equal learning capability for each expert; a finite number of items of knowledge; each problem requires certain knowledge; if the expert does not have the knowledge necessary it guesses with a probability of success, learning if it succeeds; the community choses the expert for a problem with equal probability initially, gradually biasing the choice according to success or failure; there is no communication of knowledge between experts.

Figure 6 plots: the probability that one expert will be always preferred and shows that this rapidly approaches 1.0—a best expert is determined; the expected knowledge of the better expert and that of the rival—one goes to 100% rapidly and the other is asymptotic to about 36%—there is objective evidence of the superior ability of the chosen expert. Which of the two individuals becomes the 'best expert' is, of course, completely chance.

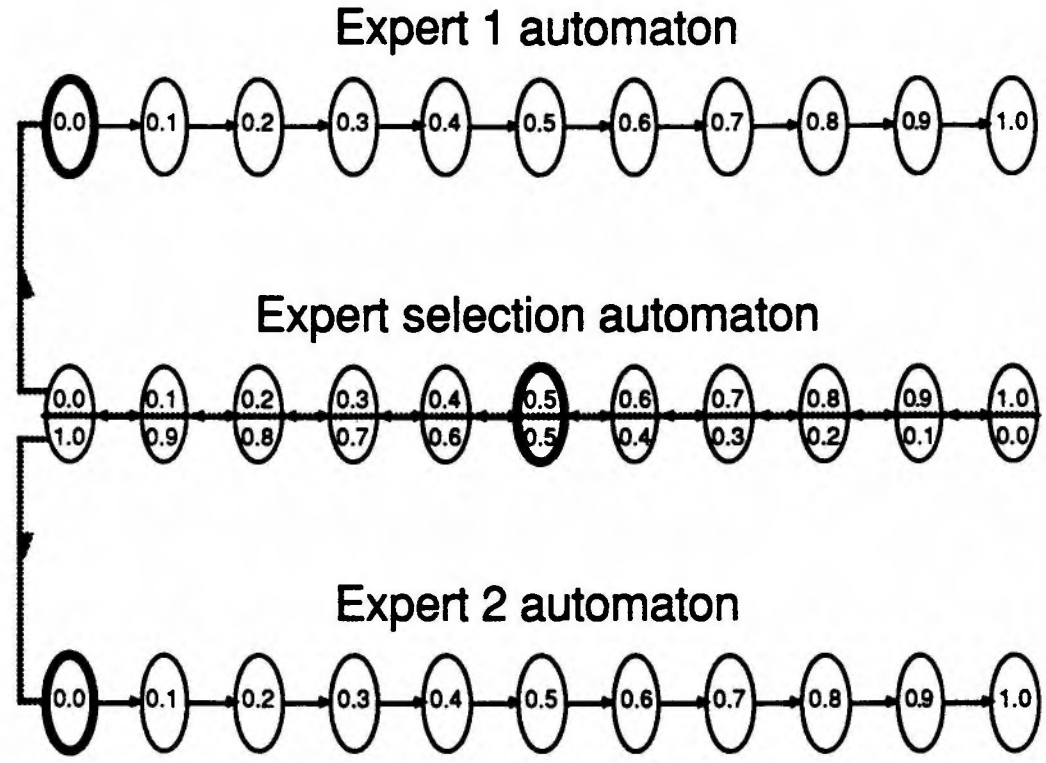
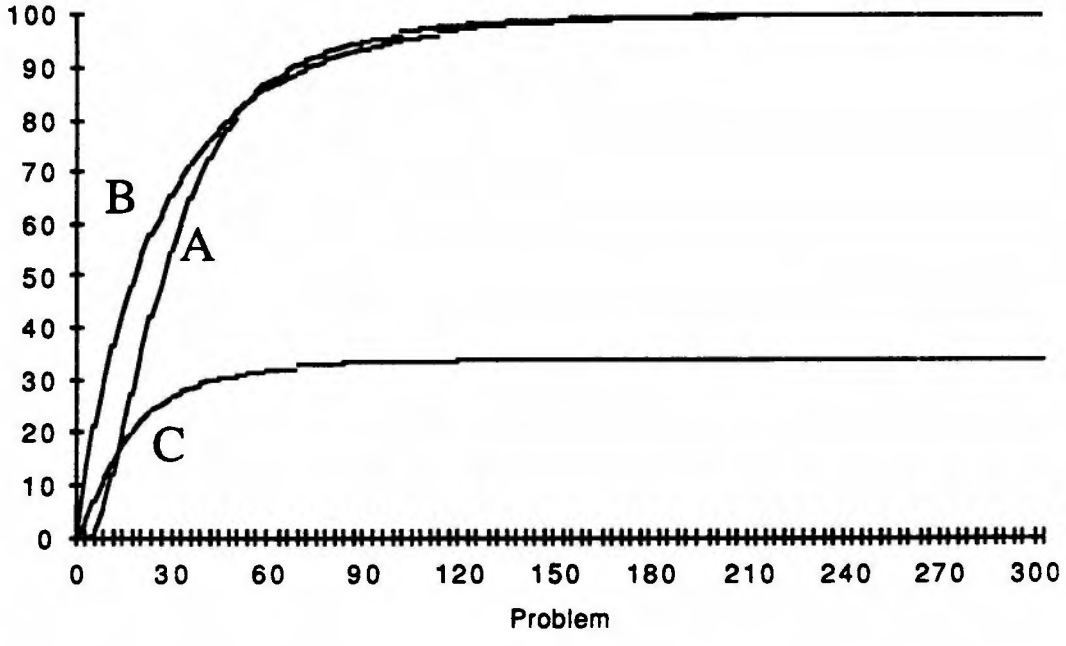



**Fig.5 Automata simulating expert selection processes**




A: Percentage probability that problem will be given to only one of the two 'experts'
B: Expected knowledge of superior 'expert'
C: Expected knowledge of inferior 'expert'

**Fig.6 The effects of positive feedback on the formation of expertise**

The simulation can easily be adjusted to take into account differences between the experts: that one starts with greater knowledge; that one learns faster; that one is favored initially (the prima facie credibility or 'well-dressed consultant' phenomenon). Similar simulations have been made of different positive feedback mechanisms, for example, if both experts are given the same problem but the problem difficulty is adjusted upwards if either gets it right—the situation of keeping up in the scientific 'rat race.' Effects have been introduced of the loss of knowledge through inadequate opportunities for its application, the growth of scientific knowledge so that there is always more to be acquired, and differential access to cultural knowledge transfer. All the simulations bear out the expected qualitative result, that a variety of different positive feedback mechanisms in society are adequate to account for differential expertise in a sample with initially equal knowledge and capabilities.

These results are not intended to suggest that individual differences are insignificant, or be a basis for egalitarian ethics. They do demonstrate, however, that the process of expertise formation is a natural systemic phenomenon in a learning society. It is functional because it distributes the processors available to different tasks through global mechanisms with no localized control. The random mechanism is efficient because it does not matter to the overall system which processors are allocated to particular tasks, and it will tend to pick out individual differences in capabilities if they exist. It puts experts into the role of traders who must offer their services to solve problems in order to gain access to those problems in order to increase their knowledge.

In practice, the professional and scientific communities also have strong mechanisms set up to reduce the effects of positive feedback in exaggerating differences, such as the ethical pressure to disseminate knowledge as widely as possible, and mechanisms for doing this through publications and seminars (Whitley 1984). However, much knowledge underlying skilled performance cannot be shared in this way (Nisbett & Wilson 1977) and the positive feedback mechanisms will always exert a strong effect.

## SOME PRACTICAL CONSEQUENCES

The popular stereotype of an expert is a fount of knowledge in some domain. If that fount can be captured in an expert system then the role of the person who previously embodied is called into account. Those outside the expert systems industry often ask the question, "why should experts put themselves out of their jobs by allowing their knowledge to be transferred to an expert system." The reply is usually that experts complain about the number of phone calls they have to answer to give advice on trivial problems. They would prefer to have the time to read journals, attend professional meetings, and think more about what they are doing. The model of expertise put forward in this paper gives theoretical foundations for this informal empirical result. The expert is not a static fount of knowledge but a dynamic process for continuously maintaining that fount by managing the acquisition of new knowledge.

The model of the expert's relation to those seeking advice as one of trading it for access to problem-solving, and hence learning, opportunities is also consistent with the actual behavior patterns of experts in relation to expert systems. Divesting the unprofitable trading to focus on the profitable is a sensible strategy, that is, passing to an expert system those standard problems where little learning opportunity is expected. However, the uncertainties about the classification of any problem situation suggest there may be dangers in this approach which could be avoided by adequate design and application of expert systems. The advice giving system should contain rules which indicate problem-solving situations that are beyond the boundaries of its capabilities and refer these to a human expert (in the same way that human experts refer difficult problems to others). It should log actual case histories of advice giving for later review by human experts so that these rules can be validated. It should make provision for feedback as to the results of advice so that learning opportunities are not lost.

The continuity between the modeling role adopted by the expert in Hawkin's model of expert-client negotiation in Figure 1, and the expert's wider modeling role in managing Goodman's reflective equilibrium in Figure 3, suggests that the performance and knowledge acquisition phases of expert systems should not be separated as they are currently. It would be better to think in terms of *knowledge support systems* that integrate knowledge acquisition and advisory roles (Gaines & Shaw 1987). In particular, the model of system-user relations as mutual negotiation rather than authoritarian advice giving is a good model for expert system.

All these considerations are also relevant to questions about the accountability of expert, expert system, and user for the advice. However, that is a side-effect of the social pressures on the inductive process—accountability, and the resultant rewards and punishments, are cultural mechanisms to encourage learning and the appropriate use of expertise. Well designed expert systems will make provision for the clear presentation of the quality of advice, reference to human experts, audit trails, logging, and case history review, because these are necessary to an effective technology—the possible legal consequences of not doing this are just a reminder of what is an effective technology in the domain of knowledge acquisition and transfer.

## CONCLUSIONS

The role of the expert in the knowledge processes of society has been analyzed and shown to be that of managing the inductive dynamics of knowledge acquisition. Experts trade the skills based on their knowledge for access to problems providing them with new material for the inductive process. The formation of expertise is functional in general because it leads to division of labor in the management of knowledge acquisition. The development of an individual expert is a random process brought about by strong positive feedback loops in the social process, for example, that a proto-expert with superior performance is brought more problems and hence has a greater opportunity to learn and improve that performance. A diversity of such positive feedback processes operate in the professions and sciences with little relation between them except their overall effect in promoting the formation of expertise. Sociological evidence for the existence of these effects has discussed, and simulation studies showing the quantitative phenomena described.

This model of the nature of expertise provides foundations for knowledge acquisition and transfer processes in expert systems and associated knowledge acquisition tools. It also places expert systems technology within a social context and provides a basis on which social and legal aspects of their use and its implications may be analyzed.

## ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the National Sciences and Engineering Research Council of Canada.

## REFERENCES

Blume, S.S. (1974). **Towards a Political Sociology of Science**. New York: Free Press.

Gaines, B.R. (1977). System identification, approximation and complexity. **International Journal of General Systems, 3**, 145-174.

Gaines, B.R. (1987a). An overview of knowledge acquisition and transfer. **International Journal of Man-Machine Studies 26**(4) (April).

Gaines, B.R. (1987b). Integrated expert system support environments. **Proceedings of AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems.** Banff (October).

Gaines, B.R. & Shaw, M.L.G. (1987). Knowledge support systems. **ACM MCC-University Research Symposium.** Austin, Texas: MCC. pp.47-66.

Goodman, N. (1973). **Fact, Fiction and Forecast.** Indianapolis: Bobbs-Merrill.

Hagstrom, W.O. (1965). **The Scientific Community.** New York: Basic Books.

Hawkins, D. (1983). An analysis of expert thinking. **International Journal of**

**Man-Machine Studies, 18**(1), 1-47 (January).
Hume, D. (1739). **A Treatise of Human Nature**. London: John Noon.
Klir, G.J. (1985). **Architecture of Systems Problem Solving**, New York: Plenum Press.
Merton, R.K. (1973). **The Sociology of Science: Theoretical and Empirical Investigations**. Chicago: University Press.
Nisbett, R.E. & Wilson, T.D. (1977). Telling more than we can know: verbal reports on mental processes. **Psychological Review 84**, 231-259.
Rasmussen, J. (1985). The role of hierarchical knowledge representation in decisionmaking and system management. **IEEE Transactions on Systems Man & Cybernetics, SMC-15** (2), 234-243 (March/April).
Rawls, J. (1971). **A Theory of Justice**. Cambridge, Massachusetts: Harvard University Press.
Schön, D.A. (1983). **The Reflective Practitioner**. New York: Basic Books.
Stich, S.P. & Nisbett, R.E. (1984). Expertise, justification and the psychology of inductive reasoning. Haskell, T.L., Ed. **The Authority of Experts**. pp. 226-241. Bloomington, Indiana: Indiana University Press.
Whitley, R. (1984). **The Intellectual and Social Organization of the Sciences**. Oxford: Clarendon Press.

# A KNOWLEDGE ACQUISITION EXTENSION TO NOTECARDS

Brian R Gaines & Maurice Sharp
Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4

**Abstract**: There is a danger that the narrow concepts of 'knowledge' currently prevalent in computer science will result in knowledge acquisition tools tools that do not allow experts to express their knowledge naturally and without distortion. Hypermedia systems provide a technology for the construction of knowledge bases that allows a much greater variety of forms of knowledge to be incorporated than has been considered in the development of knowledge bases to date. It is attractive to use such systems for the initial elicitation of knowledge from experts in a very free and open-ended fashion, and to supplement them with structured knowledge acquisition tools based on entity-attribute elicitation and object hierarchy derivation. This paper describes KC0, an extension of the *Notecards* hypergraphic system with KSS0-like structured elicitation tools.

## INTRODUCTION

In developing interactive knowledge acquisition systems for the rapid prototyping of expert systems there is a danger that the narrow concepts of 'knowledge' currently prevalent in computer science and implemented in 'knowledge-based systems' will result in tools that do not allow experts to express their knowledge naturally and without distortion. Even if hierarchical, object-oriented systems prove to give a representation of much human knowledge that is adequate for its operationalization and computer-based application, this does not imply that their data structures are a natural format for human knowledge expression. Indeed, one can argue (Gaines 1987a) that human knowledge in general is anarchical, unstructured and non-algorithmic, whereas knowledge-based system research has focused only on the well-ordered nature of some knowledge. The use of the term 'knowledge' to mean 'those aspects of human knowledge that can be formally represented and processed' is a major restriction of its usual meaning 'acquaintance with ascertained truths facts or principles' (Oxford English Dictionary).

Knowledge acquisition systems based on personal construct psychology allow experts to chose entities and develop attributes using their own vocabularies in a very free and open-ended fashion, translating this into hierarchical, object-oriented structures as as later phase (Shaw & Gaines 1986, 1987, Boose 1985, 1986, Boose & Bradshaw 1987, Gaines 1987b). However, even these systems require that experts submit to their entity-attribute structure from the start. It is easy to argue that experts soon accept this structure and find it a natural one within which to express their knowledge. However, it is known that the initial entity selection is both critical to the utility of the technique, problematic for users, and difficult to aid through the computer.

It is always appropriate in computer-based system design to take a step outside the technology and ask 'how would we perform the task without the computer?' The image will probably come to mind of the person attempting to communicate knowledge scribbling furiously surrounded by scraps of paper. We make notes on our ideas, draw sketches, re-write, re-draw, change the order of presentation, bring fragments into juxtaposition, and so on. This happens in knowledge engineering sessions—it happens with computer-based structured interviewing—and it is a computer-supportable activity.

This paper describes the initial development of a system, KC0 (Knowledge Cards Zero) that extends the Xerox Notecards system (Halasz, Moran & Trigg 1987) with some of the knowledge elicitation and analysis facilities of KSS0 (Gaines 1987b). KC0 provides a knowledge acquisition environment that supports the relatively unstructured elicitation of knowledge through a hypergraphic system as a precursor to relatively structured elicitation through entity and attribute elicitation.

## HYPERTEXT, HYPERGRAPHIC AND HYPERMEDIA SYSTEMS

The concept of a *hypertext* system as a computer-based alternative to books allowing knowledge to be stored with multi-dimensional access links can be traced back to Bush's (1945) *Memex* concept which led to Engelbart's (1963) *NLS* and Nelson's (1980) *Xanadu* systems. The advent of low-cost graphic workstations made it feasible to store and link text, graphic and pictorial material and resulted in the development of *hypergraphic* systems such *Notecards* on the Xerox 1109 Lisp machine. The availability of videodiscs and CD-ROMs has made it attractive to extend the concept to include material on computer-controlled media and talk of hypermedia systems (Young 1986, Gassée 1987).

The key concepts underlying hypermedia systems (Gaines 1987c) are that:

- knowledge is stored in a completely free and open-ended form;
- a linkage structure is embedded in the material;
- paths through the material are generated by a user following parts of the linkage structure;
- a *gateway* is a link within material to other material;
- a *perspective* is the viewpoint into the material that is built up through access to it.

A range of systems is being developed for a diversity of applications, from simple hypertext systems such as Learning Tool (Kozma & Roekel 1986) and Guide (Owl 1986), through electronic books (Weyer 1982) and encyclopedias (Weyer & Borning 1985), through programmable hypergraphic systems such as NoteCards (Halasz, Moran & Trigg 1987), through complex, multi-media systems for literary studies (Yankelovich, Landow & Cody 1987) to knowledge-based, multi-media systems for expert system development (Neuron 1986), computer-integrated manufacturing (Gaines 1986) and sports coaching (Vickers & Kingston 1987).

The hypermedium concept is significant in capturing the computer-supported, open-ended access to a diversity of media that offers so many attractive possibilities for creative thinking, problem solving and networking by individuals and dispersed groups. It provides a basis for the integration of the very wide variety of resources offered by information technology within a coherent but non-restrictive framework. Data-processing, simulation, information retrieval, text processing, presentation graphics, databases, knowledge bases, expert systems, knowledge acquisition, and so on, may all be regarded in a significant sense as particular aspects of hypermedia systems.

## STRUCTURED KNOWLEDGE ACQUISITION IN HYPERMEDIA

Hypermedia systems seem to provide a technology for the construction of knowledge bases that allows a much greater variety of forms of knowledge to be incorporated than has been considered in the development of knowledge bases to date (Gaines & Vickers 1987). In particular, they provide an interesting vehicle for knowledge acquisition since the expert can enter data in a very free form and then annotate it with a variety of cross-linkages. VanLehn (1985) has reported on a successful experiment in large-scale knowledge structuring using Notecards, and claimed that it led to the generation of structures which would otherwise have been missed. He used the programmability of Notecards to extend it with special formats and linkage structures related to the development, expression and validation of lines of argument.

We have extended Notecards in a similar fashion to include structured knowledge acquisition based on entity-attribute elicitation. Knowledge Cards Zero is intended to provide a knowledge support environment in which an expert can use:
- Notecards' data entry facilities to emulate an enhanced form of 'scribbling notes on paper;'
- Notecards' hypergraphic linkage capabilities to begin to organize the raw notes through annotation and aggregation of related material;
- structured elicitation to construe the organized material in entity-attribute terms;
- inductive analysis to derive an object hierarchy from the entity-attribute relations.

The primitive element in Notecards is a "notecard" which can contain text, graphics, any mixture of these, and gateways linking to other notecards. Special-purpose notecards are defined within the systems, such as *browsers* that show a map of the linkage structures between notecards, and *fileboxes* that group sets of notecards together. Notecards is written in Lisp and provision is made for its extension through the addition of other special-purpose notecards whose functionality is programmed in Lisp. This programmability is an important feature that is lacking on current commercial hypergraphic systems for personal computers such as the Macintosh.

KC0 is a port of the knowledge acquisition functionality of KSS0, a knowledge support system, from Pascal on the Macintosh into Lisp in the Notecards environment on the Xerox 1109/1186. The structured knowledge elicitation is implemented as a set of special-purpose notecards that provide:
- a status card showing the purpose of the elicitation and naming the classes of entities and attributes (Figure 1);
- an entities card listing the entities considered (Figure 2);
- an attributes card listing the attributes elicited (Figure 3);
- a triadic elicitation choice card prompting the elicitation of attributes (Figure 4);
- a rating card allowing attributes to be entered and entities to be rated on attributes (Figure 5);
- the rating card with entities trated on attributes by spatial click and drag (Figure 6);
- various forms of entity-attribute structure analysis such as ENTAIL allowing object hierarchies to be derived.

These special purpose notecards are currently being integrated into the full Notecards environment so that gateways may be generated: from normal hypergraphic material to elicitation notecards; and from elicitation notecards into normal hypergraphic material. This will allow users to chose entities in the normal material and link them to elicitation, and also to use Notecards to annotate the elicitation process, for example, with explanations of the meaning of attributes.

## CONCLUSIONS

An extension of Notecards has been described which allows the freedom and flexibility of hypergraphic knowledge elicitation to be combined in an integrated fashion with more structured entity-attribute elicitation leading to object hierarchy derivation. It is suggested that this combination provides an attractive knowledge support environment for experts in the rapid prototyping of expert systems. A hypermedium shell can provide a useful framework in which many other forms of more structured elicitation can be embedded.

## ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the National Sciences and Engineering Research Council of Canada.

## REFERENCES

Boose, J.H. (1985). A knowledge acquisition program for expert systems based on personal construct psychology. **International Journal of Man-Machine Studies 20**(1), 21-43 (January).

Boose, J.H. (1986). Rapid acquisition and combination of knowledge from multiple experts in the same domain. **Future Computing Systems** 1(2), 191-216.

Boose, J.H. & Bradshaw, J.M. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. **International Journal of Man-Machine Studies 26**(1) (January).

Bush, V. (1945). As we may think. **Atlantic Monthly** (176), 101-108 (July).

Engelbart, D.C. (1963). A conceptual framework for the augmentation of man's intellect. Howerton, P.W. & Weeks, D.C., Eds. **Vistas in Information Handling 1.** pp.1-29. Washington, D.C.: Spartan Press.

Gaines, B.R. (1986). The design of expert systems for planning flexible manufacturing. **Proceedings of Japan-USA Symposium on Flexible Automation.** 761-768. Osaka (July).

Gaines, B.R. (1987a). An overview of knowledge acquisition and transfer. **International Journal of Man-Machine Studies 26**(4) (April).

Gaines, B.R. (1987b). Rapid prototyping for expert systems. Oliff, M., Ed. **Proceedings of International Conference on Expert Systems and the Leading Edge in Production Planning and Control.** pp.213-241. University of South Carolina.

Gaines, B.R. (1987c). Functionality of hypermedia. **Proceedings of Hypertext'87.** University of North Carolina.

Gaines, B.R. & Vickers, J.N. (1987). Instantiation of knowledge structures in books and hypermedia. **Technical Report KSI/NML-87-BHM.** Knowledge Science Institute & Neuro-Motor Laboratory, University of Calgary (August).

Gassée, J.L. (1987). Machine vs computer. **78th Conference of Special Libraries Association**, Anaheim, California (June).

Halasz, F.G., Moran, T.P. & Trigg, R.H. (1987). Notecards in a nutshell. Carroll, J.M. & Tanner, P.P. (Eds), **Proceedings CHI+GI 1987.** New York: ACM.

Kozma, R.B. & Roekel, J.V. (1986). **Learning Tool Manual.** Ann Arbor, Michigan: Arborworks Inc.

Nelson, T. (1980). Interactive systems and the design of virtuality. **Creative Computing**, 6(11), 56-62 (November).

Nisbett, R.E. & Wilson, T.D.. (1977). Telling more than we can know: verbal reports on mental processes. **Psychological Review 84**, 231-259.

Neuron (1986). **Reference Manuals for aiScreen and aiVision.** Palo Alto, California: Neuron Data.

Owl (1986). **Guide: Hypertext for the Macintosh.** Bellevue, Washington: OWL International Inc.

Shaw, M.L.G. & Gaines, B.R. (1986). Interactive elicitation of knowledge from experts. **Future Computing Systems, 1**(2), 151-190.

Shaw, M.L.G. & Gaines, B.R. (1987). KITTEN: knowledge initiation & transfer tools for experts and novices. **International Journal of Man-Machine Studies 27**(1) (July).

VanLehn, K. (1985). Theory reform caused by an argumentation tool. **Xerox PARC Technical Report ISL-11**, 1-15. Palo Alto, California: Xerox Palo Alto Research Center (July).

Vickers, J.N. & Kingston, G.E. (1987). Modelling the master coach: building an expert system for coaching. Norrie, D., Ed. **Learning in Future Education: Proceedings of International Conference on Computer Assisted Learning in Post-Secondary Education.** pp.207-212. University of Calgary.

Weyer, S. (1982). The design of a dynamic book for information search. **International Journal of man-Machine Studies 17**(1), 87-107 (July).

Weyer, S.A. & Borning, A.H (1985). A prototype electronic encyclopedia. **ACM Transactions on Office Information Systems 3**(1), 63-88 (January).

Yankelovich, N., Landow, G.P. & Cody, D. (1987). Creating hypermedia materials for English literature students. **SIGCUE Outlook 20**, 1-11.

Young, J.S. (1986). Hypermedia. **MacWorld 3**(3), 116-121 (March).

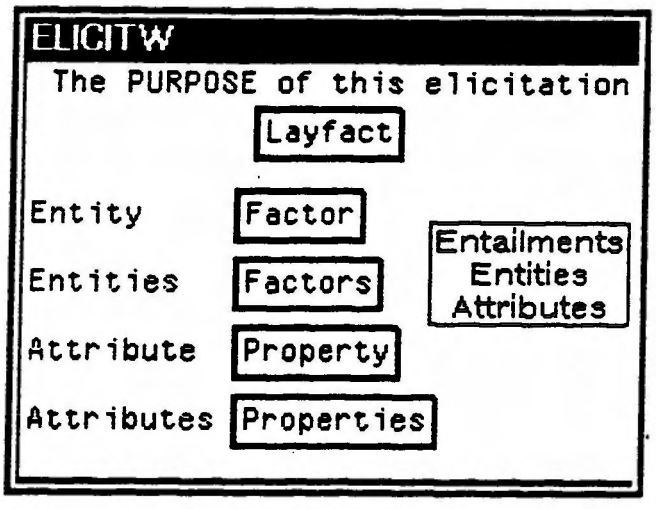



**Fig.1 A status card showing the purpose of the elicitation and naming the classes of entities and attributes —menu of options showing on right**

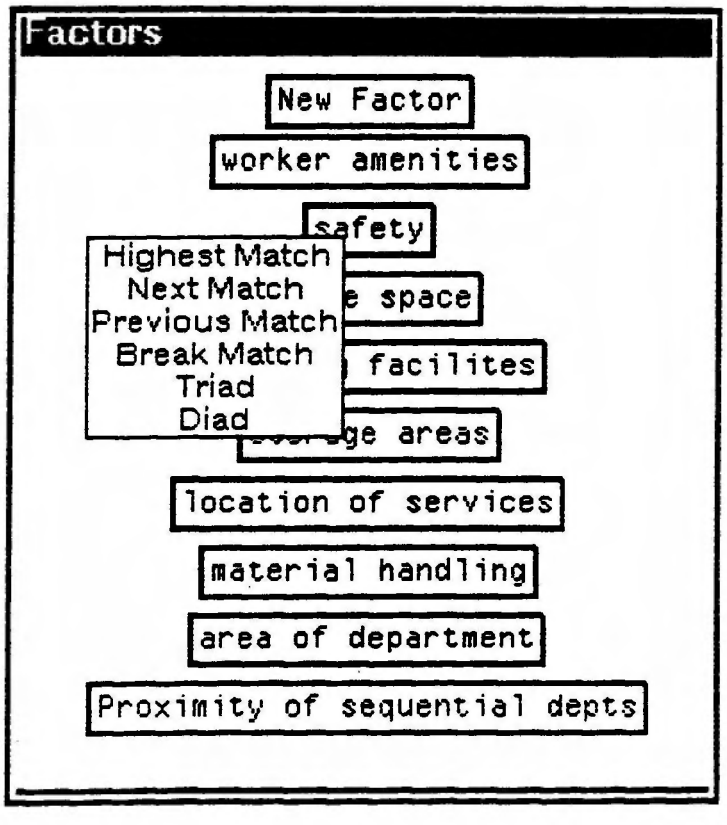



**Fig.2 An entities card listing the entities considered —menu of options showing on left**

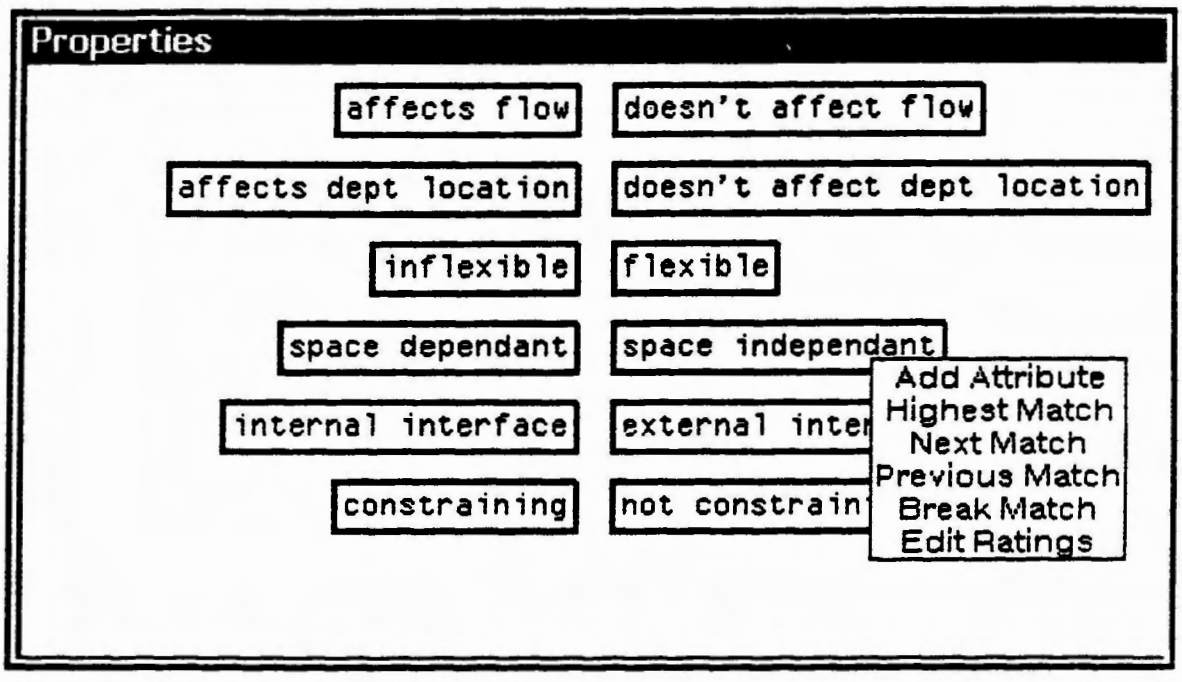



**Fig.3 An attributes card listing the attributes elicited —menu of options showing on right**

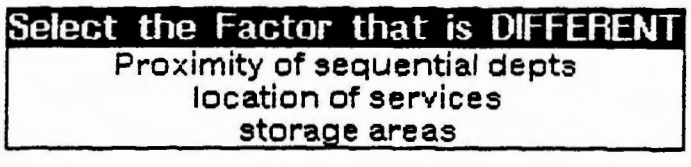



**Fig.4 A triadic elicitation choice card prompting the elicitation of attributes**

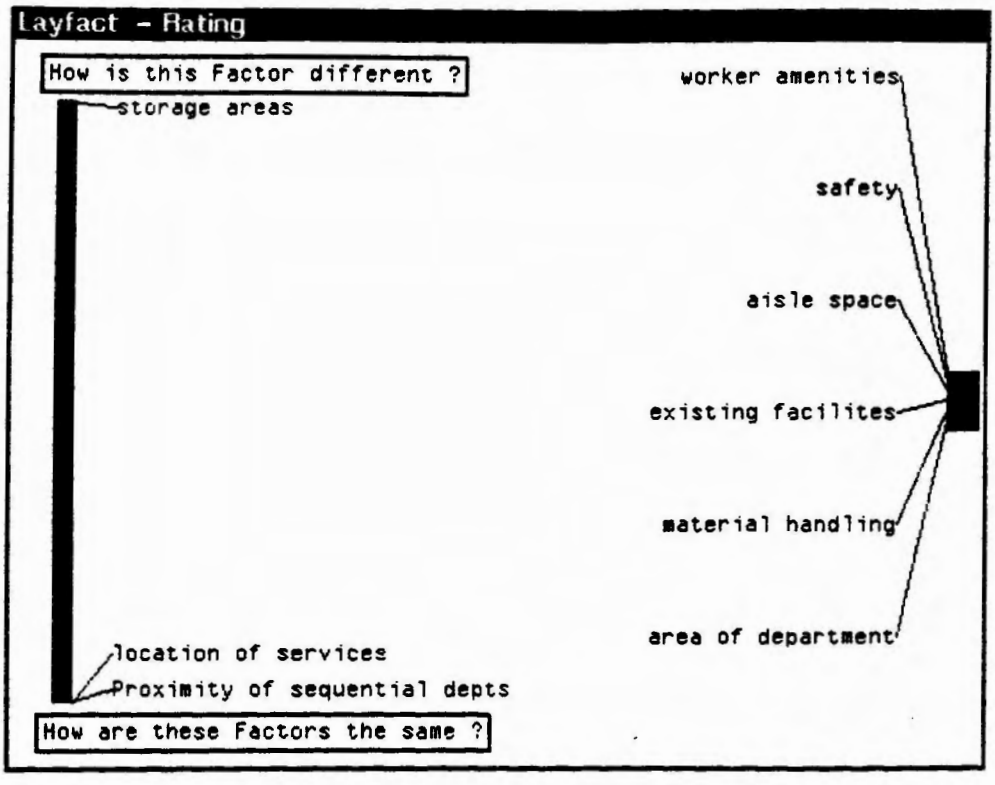



**Fig.5 A rating card allowing attributes to be entered and entities to be rated on attributes**

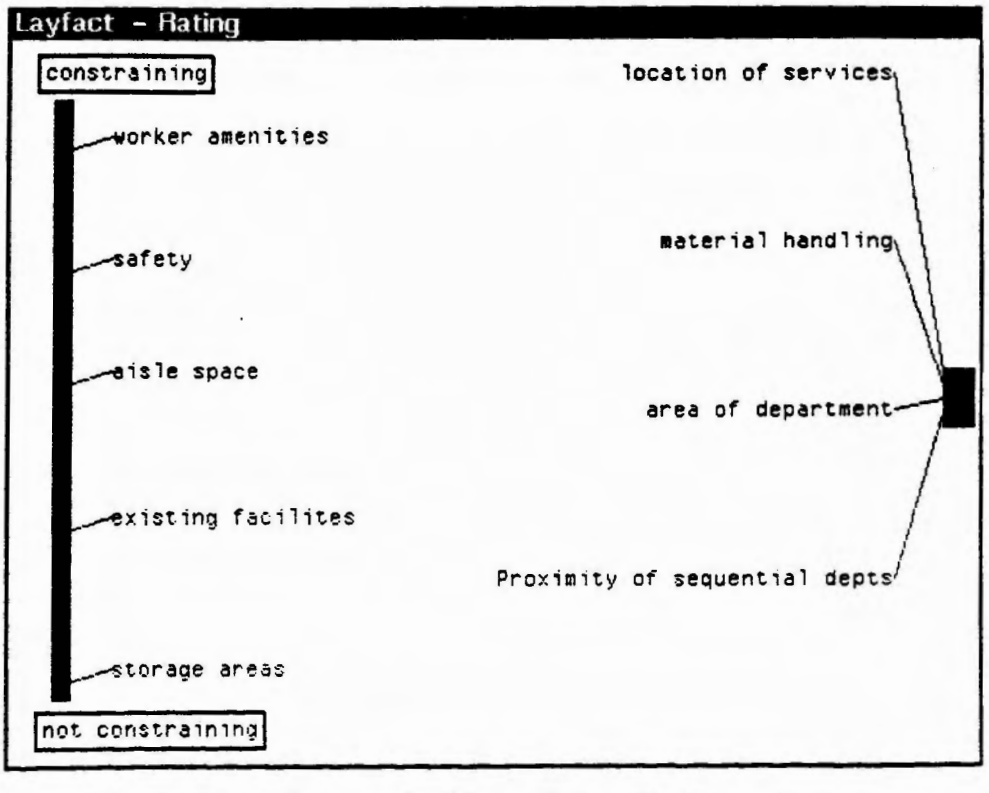



**Fig.6 The rating card with entities rated on attributes by spatial click and drag**

# Knowledge Acquisition for Real-Time Knowledge Based Systems

Bryan Clarke,
Artificial Intelligence Group,
Cambridge Consultants Ltd.

## Abstract

This paper describes the problem of knowledge acquisition for real-time (or on-line) knowledge based systems (KBS). A number of knowledge elicitation techniques and analysis methodologies adopted from the task analysis field are given. The use of these is described for two case studies: air traffic control and on-board helicopter fault diagnosis.

## 1. Introduction

In the last few years, at Cambridge Consultants, we have developed a number of real-time knowledge based systems, for example, a stockmarket equity trader assistant and an on-board fault diagnosis system for helicopters. In doing so, we have found knowledge acquisition for these systems to be a more complex process than that for what might be termed "classical" or "conventional" knowledge based systems.

## 2. Real time knowledge based systems

Numerous KBSs have been built to tackle 'static' problems, where the system is reasoning about fixed or stationary data. With these 'classical' systems, eg. MYCIN, the data remains unchanged for the duration of the consultation with the system.

There is now an increase in the number of expert systems being developed to tackle 'dynamic' problems, where the system is continuously reasoning about constantly changing data. This type of system is generally referred to as a real-time KBS (other descriptions are on-line KBS, embedded KBS or continuous reasoning KBS). This type of system is being used to carry out monitoring and control tasks, such as process plant supervisory control or on-line fault diagnosis as described by Bennet [1] and Cartwright et al [2].

## 3. Difficulties presented in real-time KBS knowledge acquisition

Knowledge elicitation and acquisition are vaguely defined in the literature and require some clarification before proceeding. I have adopted the definitions from Breuker et al [3], which defines knowledge elicitation as the methods and techniques used in abstracting knowledge from a human expert or specialist in the domain of interest. Knowledge acquisition has a similar definition, but includes sources other than the expert, eg. books, manuals, users, etc, and the concept of rapid prototyping.

Knowledge acquisition has often been referred to as the bottleneck in implementing KBS, as codifying knowledge held by an expert or document into a machine-usable form is an extremely difficult process. The problem is increased for real-time KBS, because effectively another dimension is added to the problem, that of time. In developing a real-time KBS, the dimension of time must be taken into account when acquiring the knowledge and building the system, because it usually contributes to the problem and the methodology used by the expert in solving it. Also, the time factor may present difficulties for the knowledge engineer in gaining access to the expert for interviewing and observation.

## 4. Temporal knowledge and resource limitations

In developing real-time KBSs we are trying to embed an expert's continuous reasoning capability as well as their domain knowledge within a program. Also, we need to determine the dynamics of the problem the expert is solving, eg. the data rate in controlling a process plant. During elicitation, the following issues have to be addressed:

(1) Acquisition of the expert's temporal domain knowledge, such as:

* trends and histories used by the controller in solving a problem.
* expectations and predictions used by the expert in solving a problem.
* event sequences.
* data validity.
* plan execution durations, eg. in process control, how long does it take the process to respond and stabilise after a perturbation by the controller.

(2) Acquisition of the expert's temporal reasoning (control) knowledge, such as:

* handling of concurrent subproblems.
* handling of sequential subproblems, ie. scheduling knowledge.
* handling of interrupts, ie. shelving reasoning.

(3) Determining the expert's resource limitations and how he/she copes with them. For instance:

* focusing, eg. in a complex plant, the control room operators cannot scan all the dials at once, so they tend to employ a focusing mechanism where they scan sets of dials at regular intervals.
* data prioritisation is often employed by an expert when faced with high data rates, eg. nuclear power station emergencies involving hundreds of alarms being tripped.
* memory limitations, ie. data retention and discarding.

### 4.1. Elicitation techniques

There are numerous elicitation methods for conventional KBSs as described in Breuker et al

[3], Burton et al [4], and Clare et al [5]. These are perfectly applicable for developing real-time KBSs, but we have supplemented these methods with a number of ad-hoc elicitation techniques when carrying out elicitation sessions as described in the "Case studies" section. We cannot claim to have found any formal methodology, but the techniques are worth describing:

Walk-through of a problem snap-shot

In cases where the task and task environment is frenetic, eg. share trading in the stockmarket, it difficult to comprehend the task being performed by the expert, because of the large volume and rate of data being handled. In these cases to appreciate the task, it is useful to take snap-shots of the environment and ask the expert to describe (or walk-through) how they were performing their task at the time of the snap-shot.

Observing the expert in the real task environment

Even though the task being carried out by the expert may only be partially comprehendable, this technique provides a valuable insight into:

* the sub-tasks making up the main task.
* the time constraints and resource limitations of each sub-task.
* data required for each sub-task.
* communications made during each sub-task.

It may, however, be impossible to observe the expert, eg. pilots in a single-seater aircraft.

Observing the expert in a simulated task environment

Where it is impossible to gain access to the expert even for passive observation for safety reasons, eg. air traffic controllers, or because it is physically impossible, eg. pilots. The only possible access to the expert may be in a simulated task environment, eg. air traffic control simulator or flight simulator. This has the disadvantage of generally being an inexact replica of the real environment, but it does offer some benefits:

* Events can be injected into the environment to find the expert's response, unlike the real world.
* During intensive activity, the simulator can be slowed down or stopped so the expert can be questioned.
* During uneventful durations, the simulator can be speeded up to reduce time wastage.

Shadowing the expert by a second expert

In cases where a task is time critical, interrupting the expert with questions can disrupt the task being carried out. One approach to overcome this is to ask a second expert to explain the reasoning and actions of the expert performing the task. In some cases this may result in problems as experts do not always perform or reason about the task in the same way.

Simulator development

Often in developing and testing a real-time KBS, a data generator or simple simulator of the process being reasoned about has to be developed. A by-product in the development of this simulator is education of the knowledge engineer in the domain.

Role reversal
The knowledge engineer and the expert reverse roles such that the expert is the mentor and the knowledge engineer is the student interpreting or controlling the process. This gives the knowledge engineer an appreciation of the tasks involved, time and resource limitations and the data and knowledge the expert calls upon for each task.

## 5. Knowledge analysis techniques

We found a number of techniques borrowed from task analysis field, as described by Barnard et al [6], useful for analysing and documenting temporal knowledge:

The process and information flow model
In order to complete a task, a number of subtasks or processes must be completed. The order these processes are executed is often dependent upon information resulting from a previous process or information changes generated by the task environment. The approach here is to decompose the task into inter-dependent processes with information flowing between them. This generates a computational model of the task with the results being presented in a flow chart or a program pseudo-code representation.

The goal structure model
Usually a plan of action is required to carry out a task, whose form is dependent upon the goals or objectives to be achieved. Thus one approach to analysing a task is to extract the goals to be fulfilled and where possible decompose these goals into a number of subgoals. This analysis results in a hierarchical structure of goals and subgoals representing the plan of action. In monitoring and control tasks, there are often concurrent goals that are prioritised by the expert. The prioritisation process can be captured using this technique.

The information structure model
This analysis builds on the goal structure analysis. Each goal (or sub-goal) is examined to determine the information (data and knowledge) needed to achieve the goal and the information produced by the goal (the goal products). The products are divided into those output by the expert which change the task environment (external products) and those which change the expert's cognitive representation of the task and environment (internal products). This technique extracts the dependencies of goals on each other and on the task environment, eg. goal X uses products from goals Y and Z and data D from the environment, thus goals Y and Z must be satisfied and data D must be present before goal X can be satisfied.

## 6. Case studies

The previously described knowledge elicitation and analysis techniques have been used in two cases: air traffic control and helicopter on-board fault diagnosis.

## 6.1. Air traffic control

As part of a study into assessing the feasibility of applying real-time KBS techniques to air traffic control (ATC), we carried out experimental knowledge elicitation sessions to determine whether controller knowledge (or expertise) could be captured†. The controllers carry out a complex monitoring and control task in an intricate task environment, thus knowledge acquisition is a difficult problem. As it is not possible to observe or interview controllers whilst they are controlling aircraft for safety reasons, the problem is increased. Even so, we applied some of the techniques described to elicit knowledge:

* We interviewed controllers at the air traffic control centre on a stand-by suite (used as a backup), but not controlling aircraft. A suite consists of a radar display showing the current airspace picture, various communication facilities and data displays. This provided a useful insight into the controller's job, but it was difficult to elicit knowledge because of the high air traffic flow rates. Also, we wanted to observe controllers tackle situations generated by ourselves, eg. various weather situations and emergencies, at a rate we could comprehend. We thus decided to give the controllers snap-shots of the task to handle.

* The snap-shot scenarios were generated by a controller. They consisted of various air traffic situations for a sector of airspace and were of increasing complexity. The reasons for generating a scenario of a particular form also fed into the knowledge elicitation results.

  The controllers were asked to describe the scenarios and comment on how they would handle the situation. During this we questioned them to obtain a further insight into their approach to the situation. We injected further constraints into the scenario, such as emergencies or adverse weather conditions. All these sessions were videoed for later analysis.

* We observed and videoed controllers handling simulated aircraft on the TCSDG (Terminal Control Systems Development Group) facility at the Royal Signals and Radar Establishment. This provides a good approximation to a controller's working environment. The TCSDG consists of an air traffic simulator interfaced to a number of controller and pilot consoles. A synthetic radar display shows air traffic, navigational beacons and geographical data. The controller issues commands or receives requests from simulated aircraft.

  As two controllers were available, it was possible to have one controller handling aircraft and the other explaining his actions. This turned out to be infeasible, as the shadowing controller had to point at the radar display which distracted the other controller.

  We tried handling aircraft on the TCSDG facility with the controllers acting as our mentors. This proved to be extremely useful in giving us an appreciation of the factors taken into account when controlling aircraft.

---

†This work is being carried out under contract to the Civil Aviation Authority.

To analyse and document our findings from these sessions, we applied the three knowledge analysis techniques previously described. Some of the results are shown in Figs 1-3.

### 6.2. Helicopter on-board fault diagnosis

Some of the described elicitation techniques were also employed in developing an on-board fault diagnosis system for helicopters*, see Clarke [7]:

* After reading literature on the helicopter's operation (the "aircrew manual") and potential faults (the "helicopter fault flip-charts"), we observed pilots in a sophisticated helicopter simulator. The simulator provides a replica of the helicopter cockpit and simulates the cockpit view for a number of flying environments, eg. night flying over coastal areas or daylight flying over hilly terrain. The simulator instructor injected faults into the simulator and we observed the response of the pilot (the pilot's diagnosis and recovery actions). It was not possible to directly question the pilot as this distracted him from the task, but he gave a constant commentary of his approach to handling a fault. After the simulator session, we interviewed the pilots to obtain clarification on points we did not understand.
* In order to develop and test the on-line fault diagnosis KBS, we wrote a simple helicopter simulation to provide data to be reasoned about. This proved to be fruitful in eliciting further knowledge from the pilots. When this simulator was demonstrated to the pilots they were able to point out errors or discrepancies with the real helicopter providing us with a greater insight into their diagnostic reasoning.

## 7. Summary

In developing real-time KBSs, the knowledge acquisition techniques of conventional KBS need to be supplemented so that the expert's temporal knowledge and continuous reasoning ability can be embedded in the system. This paper has described a range of techniques which were found to be useful for eliciting and analysing this knowledge borrowed from the task analysis field.

Distributed software specification methodologies, such as SREM, see Alford [8], also provide useful techniques for real-time KBS knowledge acquisition. For example, SREM allows the concurrent as well as the sequential processes to be represented explicitly. We hope to investigate these techniques in the future.

## 8. References

(1) Bennett, M.E. *Real-time continuous AI systems.* IEE Proceedings, Pt.D, Vol.134, No.4, July 1987.

---

*This work has been carried out under contract to the Human Engineering Division, Royal Aircraft Establishment.

(2) Cartwright, C and Ruskin, P. *Musings on the needs of real-time AI toolkits*. Expert Systems User, Vol 2, No 9, Dec 1986.

(3) Breuker, J.A. and Wielinga, B.J. *Analysis techniques for knowledge based systems*, part 2, Methods for Knowledge Acquisition, Report 1.2 Esprit Project 12, University of Amsterdam, Dec 1983.

(4) Burton, M. and Shadbolt, N. *Knowledge Engineering* in *Systems for users*. N.Williams and P.Holt (editors). McGraw-Hill. To be published.

(5) Clare, J.N. and Young, R.M. *A review of knowledge elicitation methods*. Esprit Project PP106 - KAI Part II, July 1984.

(6) Barnard, P.J. and Harris, J.E. *A feasibility study of tools for representing operators' interactions with naval command and control systems*. May 1984, Scicon Human Factors Report HR-2.

(7) Clarke, B. *On-board fault diagnosis for helicopters: an AI approach*. To be presented at MilComp 87.

(8) Alford, M. *SREM at the age of eight: the distributed computing design system*. IEEE Computer, April 1985.

```
BEGIN handling aircraft
    (evaluation of scenario)
    identify classes of aircraft (see N1)

    FOREACH class of aircraft
        FOREACH aircraft in cluster
            path prediction from current heading & flight plan
            identify problems from predicted path (a/c run-
                ounds, boundary problems & conflict situations.
            mental note of problem (see N2)
        ENDFOREACH
    ENDFOREACH

    prioritise problems according to urgency (see N3)
    select most urgent problem
    bound problem (see N4)

    (sorting problems out)
    FOREACH aircraft in problem
        identify intentions of aircraft (see N5)
    ENDFOREACH

    FOREACH aircraft in problem
        generate plan options for aircraft (see N6)
    ENDFOREACH

    do
        select 'best' plan option (see N7)
        test plan option (see N8)
        IF plan fails
        THEN note reasons
        ENDIF
    UNTIL acceptable plan obtained OR no plan options left

    IF no plan options left (see N9)
    THEN select the 'least' of a bad bunch
    ENDIF
END
```

Fig. 1: ATC - Process model for handling aircraft

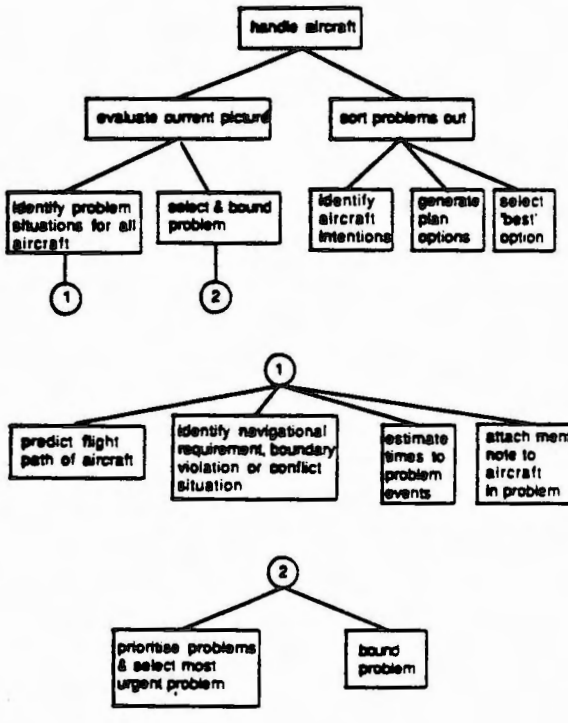



Fig. 2: ATC - Goal structure for handling aircraft

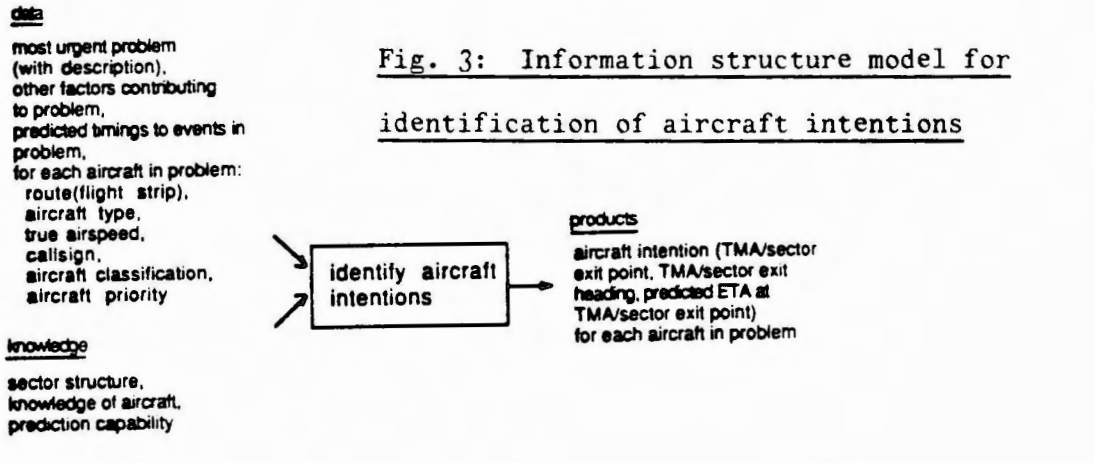



Fig. 3: Information structure model for identification of aircraft intentions

C3

# Knowledge Acquisition for the Advice System

Elizabeth S. Cordingley

Research Fellow, Department of Sociology,
University of Surrey, Guildford GU2 5XH, England
(JANET address: Cordingley@UK.AC.SURREY.SYSE)

January 1987

Acknowledgement
This work was carried out as part of the DHSS Demonstrator project, supported by the Alvey Directorate of the UK Department of Trade and Industry and the Science and Engineering Research Council. The project collaborators are ICL, Logica, Imperial College London and the Universities of Lancaster and Surrey. The support of other project members is gratefully acknowledged. The views expressed here are those of the author and may not necessarily be shared by other collaborators.

## Introduction

Initial acquisition of knowledge for the prototype Advice System being developed at the University of Surrey as part of the Alvey DHSS Demonstrator Project has taken three forms. Sources of knowledge were identified and the knowledge about Child Benefit, the initial focus of the system, was extracted. Intermediate representations of portions of the knowledge were created. The knowledge was represented in Prolog. This paper gives brief details of the three processes

## Extracting knowledge from public sources

A number of documents contain information and guidance about UK Social Security Benefits. The National Association of Citizen Advice Bureaux (NACAB), and the Department of Health and Social Security (DHSS) are prominant among those who prepare aide-memoire, guidance manuals, and circulars to keep advice workers up to date and to clarify issues These are generally unavailable to members of the public.

This exercise was based on publicly available documents. There are publicly available texts such as the Penguin Guide to Supplementary

Benefits by Tony Lynes which were not used as they contained little or no information about the benefit in which we were interested. Child Benefit. Another which might have been used but was not was the Guide to the Social Services produced by the Family Welfare Association. It is published every two years and was in revision, due for publication in March '87. The previously available version was out of print and a more out of date that was felt desirable. Two other books, identified as being primary documents used by advice workers are revised annually and it was from 1986 editions of these that Child Benefit knowledge was taken for this exercise. The first source, one of the Child Poverty Action Group (CPAG) handbooks rights guide to non-means-tested social security benefits, is widely used by Citizen Advice Bureaux (CABx) advice workers and those in other independent adivce centres. The second source, Tolley's Social Security and State Benefits has DHSS sanction and is used by its Regional Freefone service. DHSS leaflets and material contained in the back of Child Benefit pay books was also a publicly available knowledge source

Other material in the public domain included the Child Benefit Act 1975 and its associated statutory instruments, primarily Regulations and their Amendments. These were to be the foundation of later work and were not referred to even during the writing the Prolog rules. Where necessary, guesses were made about details not contained in the intermediate representation in order to produce a more complete Prolog program.

The information contained in Tolley's and the CPAG handbook regarding Child Benefit is substantially the same, but the writing style and the order in which they present information and advice is slightly different. Though both texts give references to the legislation, Tolley's does not cite reported Commissioners' decisions while the CPAG handbook does. This reflects the expectation on the part of CPAG that the handbook will be used to build a case for the claimant to use in writing to the DHSS or in an appeal. DHSS advisors do not build cases and so perhaps are not hampered by the fact that Tolley's would not be as useful for doing so. Two extracts below concerning the meaning of 'responsibile for' a child illustrate these sources. Not all of this knowledge was included in the initial intermediate

representation and comparison of these extracts with the diagrams would indicate some of the work on this knowledge yet to be done.

The two excerpts are reproduced as faithfully as possible including spacing, italic and bold faced type

CPAG (Section 5 1 (v) (b), p 91)

**(b) 'Responsible for'**

This means that

*either* (a) the child is living with you

'The child must live in the same house of other residence as the parent and, moreover, be carrying on there with the parent a settled course of daily living'.[23] This does not mean, however, the same as 'residing together' or 'presence under the same roof'[24] (see p93). A child may be 'living with' someone even while temporarily away. An absence for any reason of up to 56 days in 16 weeks will be ignored.[25] Thus, even where entitlement ceases because, for instance, the child is taken into care, benefit is payable for the next 8 weeks A child will still be treated as living with you while receiving full-time education at a boarding school.[26] You will keep child benefit for a total of 20 weeks when a child goes into hospital or into certain other forms of residential accommodation for medical purposes. After this period, the child will still be treated as living with you only if you are regularly incurring expenditure in respect of him or her as below;

*or* (b) you are contributing the current rate of child benefit per week (ie, £7 00), for the maintenance of the child.[27]

The notes refer to reported Commissioners' decisions and statutes. Note 23, R(F) 2/8, is a reference to the 2nd reported decision in 1981 relating to Child Benefit and family allowance. Note 27, s3(1)(c) CBA 75, is a reference to section 3, subsection 1c of the Child Benefit Act 1975. These notes are at the back of the handbook. References included in the passage were the following:

23 R(F) 2/81
24 R(F) 2/79
25 s3(2) CBA 75
26 s3(3)(a) CBA 75
27 s3(1)(c) CBA 75

Tolley's (Section 6.37, p 107)

6.37 *Person 'responsible' for child* A person is regarded as responsible for the child if he or she

(a) has the child living with him or her during the week concerned; or

(b) is contributing towards the child's maintenance at a weekly rate at least equal to the rate of benefit.

A person who has had a child living with him or her will be regarded as still having the child living with him or her if the child is away, provided that

- the child is away for less than a total of 56 days(eight weeks) in a period of 16 weeks; or
- the child is at boarding school; or
- the child is in hospital or a local authority home for the physically or mentally handicapped (but this rule only applies for the *first twleve weeks* of the stay in the hospital or home; after that the person will only be regarded as still having the child living with him or her if regularly incurring expense by way of pocket money or small gifts for the child, the cost of visiting the child etc.)

[Child Benefit Act 1975, s 3, CB General Regs 3, 4]

There are no notes at the end of the chapter or end of the book to which the reader must refer In some parts of the text there are references to other sections of Tolley's and occassional references to one of the appendices or tables at the back.

## Intermediate knowledge representations

Adaptations from systemic grammar network notation:

An intermediate representation is needed to document the work of the knowledge analyst/engineer making it sharable and available for scrutiny. The aim is to create a single style of representation which would be meaningful to both domain experts who have no experience of writing computer programs and system builders programming the system. Systemic grammar network (SGN) notation (Butler, 1985; Bliss, Monk and Ogborn, 1983) seemed a useful starting point for developing such a representation, but little of SGN's full range of possibilities has been used so far. Details of the development will be the subject of a separate report when the work has progressed further.

The main import for this paper is that it is possible using systemic grammar notation to represent AND/OR trees. The 'Bar', -[, indicates the 'exclusive or', often written XOR, relationship i.e. that only one of the mutually exclusive categories on the branches will apply. The 'Bracket', -{, is used to indicate the AND relationship One value for each of the categories must be chosen for each case. Because of the difficulty of drawing brackets using the software to hand, and the fact that it was hoped to avoid the necessity of explaining special symbols, only bars symbols were used. The word 'or' or 'and' were put on the bars to indicate the relationship being represented. This labeling meant that someone 'reading' the diagram would not have to remember the meaning of different symbols.

Figure D1 is a high level representation of a portion of the rules relating to eligibility for Child Benefit. The portion relating to conditions which determine whether a child is or is not a 'qualifying child' were developed in more detail and is shown in Figure D2 and D3.

Initial trials indicated that non-programmers found it difficult to 'read' Figure D2 which is very similar to a straight forward AND/OR graph. This may be partly due to the fact that trial 'readers' found it difficult to think in general rather than specific terms. They were not domain experts and did not take a 'broad brush view' as domain experts might be expected to have done. The comments of trial must be interpreted with caution but were still very helpful at this stage of the work in pointing out gross shortcoming of the diagrams.

None of the representations have been tried on domain experts, either DHSS staff or experienced advice workers, although this is included in future plans. With the limited trials that have been conducted Figure D3 was easier for non programmers to follow, but remains an empirical question as to whether domain experts in this field read the diagrams differently

Trying out intermediate representations on a colleague:

Various versions of the figures were tried out on three colleagues at the University which led to devising Figure D3. It revised diagram makes the goal more obvious and shows the quick way through the conditions which will apply to a large majority of people getting Child Benefit.

The following is a transcript of what a colleague at the University made of Figure D2. It may be relevant to note that the colleague was not a member of the team. It is clear that there were a number of difficulties with the representation, especially where there are negations. The transcript, though not proofread for spelling errors and punctuation, is as near as possible to a verbatim account of what was said. My words are in parentheses ( ), and notes about the tone of voice, interruptions or tape counts in square brackets [ ]. The rest are words said by my colleague. Spaces are indicative of short, but noticable pauses. Segments of the transcsript are numbered to facilitate reference to them. As far as possible, the three main conditions which appear on the diagram are in single quotes when they are mentioned. Other details read from the diagram are in double quotes

The whole transcript is included to give a flavour of the exercise and the kind of insight about our work that can be gained before formal trials or experiments are undertaken. The insights must be taken for what they are, however, comments from colleagues not typical potential users. Segment 9 suggests that it is important that diagrams use terms the reader associates with a particular age group, e.g. 'borstal or other detention centre' not 'prison'. Segment 26 and 32 suggests the usefulness of indicating inclusion rather than exclusion conditions. In S40 a common sense term, 'school holiday', is suggested.

Transcript of university colleague 'reading' D2

S1 Child is a qualifying child which means that the child satisfies three kinds of qualifying conditions. Right, 'Residence' So I'm starting by reading "is resident in great Britain"

S2 (I explain about the asterisk and the dotted box)

S3 Ok, well I read "is resident in Great Britiain " and I say to myself, yes, say, if I'm imagining a particular child, right? (mm )

S4 "OR", but having said yes I wouldn't want to read that (ok) presumably but just for the sake of this I will read it.

S5 "Or is in Great Britain" (mm) "AND the claimant is one of those three things" [i.e. parent, step parent or adopted parent]

S6 Ok but having satisfied that then I would go on to 'Responsibility Care and Charge'. "is the responsibility of " [said as though it were a question rather than a condition] "is in the care of claimant" [ said as a condition] alright,"AND" presumably "is living with the claimant" "or is not living with the claimant" "and is being maintained by the claimant", I'm getting confused [after a significant pause continues in a much lower voice, as if thinking to herself], alright so we're talking about the child here. [another pause and even lower voice then in a normal voice] OK, [pause while reading to herself] I'm just Because these things don't apply I'm just trying to imagine what would happen if they did apply, but basically I'm not reading them that closely because I know they don't apply. So I suppose if you take an extreme case [pause]

S7 Look what is this "not in prison"? that would be [pause] the parent is not in prison [interruption from someone else in the room with a joke about what she has been saying and there is general laughter]

S8 ("Now these are all things to do with the child" pointing to the 'responsibility/ care and charge' box)

S9 So the child is not in prison. (uh hmm) The child wouldn't be in prison. It would be in care. It would be in some other sort of thing. It wouldn't be prison though, it would be borstal. [other person contributes a comment about where children are detained...]

S10 Right, so we've satisfied he's a resident and is the is responsible Is this I'm not quite sure if this is an AND statement, here [ pointing to the direct lines into the bars]

S11 (In fact these direct lines mean that this is further elaborated by that, which is further elaborated by that. So it would mean that 'Responsibility /Care and Charge' , by which one could mean "is the responsibility of the claimant", by which one could understand the the person "is living with the claimant" OR is "not living with the claimant AND is being maintained by the claimant" and "being maintained by the claimant" is understood as being not in prison and not in the care of the Local Authority)...

S12 OK, so these are situations where they are not living with the claimant but are being maintained by the claimant. OK

S13 (In other words specifically not in prison and not in..)

S14 Those are the only two situations where that Ok, fine, because they are negative

S15 OK right so lets go back Right you start off by saying its under 16 (right this is the 'Age and Educational Status' section [said for the benefit of the audial record])

S16 Then we have the complicated where they're 16 - 19 What happens there?

S17 (16 to 19 AND)

S18 Oh "AND" Sorry right," AND in full time education" (uh hum)and that full time education could be "at school" or "at another educational establishment". If it is not school then the education is either for "education is NOT for a degree, NOT [pause considered in a very low voice]

S19 If they're 16 to 19 (uh hmm) and they're in full time education but it is not for a degree. So if they are doing a degree I think no I wouldn't go down there So what happens if they are doing a degree?

S20 (Then they wouldn't satisfy one of these qualifying conditions)

S21 I see. God. OK

S22 (So if you start up at the top of this then "is less than 16 years old")

S23 Well less than 16 they automatically qualify. (right)

S24 It's just a question of understanding whether they qualif- what- wh- wh- you're actually what the goal is which is whether they qualify or not (right)

S25 I didn't realise that. So that once children go to University, they don't qualify (that's right) even if they're (for Child Benefit) for Child Benefit [musingly] Oh, well I wouldn't know this OK. (I say something which is inaudible below her next question)

S26 How how would they qualify? What else is there? How would they qualify then? If they're in edu if they're in edu Its only if they're at school, is that the case? Or if they're su

S27 (No they can be in , If they are 16 to 19 , they can either be in full time education )

S28Yes (or they can have left school, but let's just take the full-time education part)

S29 Yes (they can either at school or they can be somewhere else)

S30 Right (If they are somewhere else you have to look into it in a bit more detail)

S31 Okay (The education mustn't be any of those things. It mustn't be for a degree or for a- )

S32 But what else could it be, that's all I'm asking What could it be that would qualify?

S33 (O' Level or A'level at a Further Education )

S34 Oh, I see. So they could be at technical college. (yeah) Ok.

S35 I didn't realise, actually, that Child Benefit stopped, as I said, when you went to University but then I wouldn't know. (Well you move onto Student Grants)

S36 But what happens if you don't get a grant, [door closes noisely] because its not means tested at all Ok well I understand that, but uh so let's say that ... but if its- but if- but if it is a degree then you don't bother to look at what the financial support is (That's right) But if it is let's say it is technical college I think it would be easier to think through this if you had specific examples to work thorough. Then you'd test whether the cases apply. Ok let's say technical college doing A'Levels or something and the "financial support is" "Not from employer and not from the manpower services commission" Ok right ok ok but let's say they are not in full time education so they've left school this term (uh hum ) "AND [she laughs in desperation?] And " this is getting complicated and What's this week of claim?

S37 (the week of the claim that you're talking about, because you're talking about a child whose claiming benefit [I should have said for whom you are claiming benefit and I'm picked up on it in her next statement] and we're now talking about the week in [should have been 'for' ] which the claim is [paid] you get paid weekly) But it would be a parent that's reading this and claiming it Would they know? Presumably they would know what week of claim means

S38 (Well this wouldn't actually be) yes. OK, alright so

S39 (The week that we're talking about) Alright the week is not a week in which the child was in full time employment OK, What is this? Is it an example of full time "OR had a place on a " Oh, I see "OR had a place on Youth Training Scheme OR was receiving education with financial support from an employer". Ok so basically is when child is totally unemployed "AND" before the first Monday of the next school term" [This is actuall incorrect on the diagram which should have made reference to the three 'fixed dates', but I din't realise that at the time] [laughs] OH, God [laughs] UHm It's complicated isn't it?

S40 Uhm Ok, so thinking of a hypothetical example. Your child has left school at the end of the summer term and this is the first week ater the beginning of the summer holidays and you go in and its not a week in which they've done any of that. And it's before the first Monday Ok and it's before the next term. So its the school holiday! [pause] So if your child leaves school at 16. Is that 16 exactly [pointing to the < symbol] ? (It's less than) It's less than, so 17 is when it stops (Well on your 16th birthday) It stops on your 16th birthday. (that's right) OK Say say your child has left school at 16 and therefore doesn't, is not eligible under that (Is not less than 16) Alright is not less than 16. But they've left school summer term and they're not starting well they've got nothing So you could still claim Child Benefit up until the first Monday of the Autumn term.

S41 (Right. Or if a child left school at Easter you could claim Child Benefit through the Easter holiday, up until the first Monday of the Summer term)

S42 Oh, I see. But the minute the Autumn term, or the next term starts, then you can no longer claim it. (right) unless they go back to school or they go to further education, technical college or something like that. (Full time for doing school type studies) OK.

S43 (Right, that's very useful [tape counter now reads 129. The rest is a discussion of the diagram and ways of improving it])

## Child Benefit details in PROLOG

In order to see if the intermediate representation could easily be utilised for knowledge-base building, it was used as a source of knowledge for writing rules in LPAProlog on the Macintosh even though it is unlikely that PROLOG will be of the prototype or the final system. Not all the relevant knowledge had been incorporated into the diagrams by the time the exercise was undertaken, so it was necessary to make stop-gap guesses in order to get a sufficiently complete program to test. We also were aware that there were inaccuracies in the representation at that stage. These inaccuracies did not invalidate the exercise, but the resulting program would, of course, also be incorrect.

The process of writing the Prolog program took approximately four hours and the program ran successfully first time. Some work had been done by my colleague recently in Prolog and this was freely used in the program . This is noted where relevant. Although two of us were working on it, one person familiar with both PROLOG and the intermediate representations could have accomplished as much in virtually the same time.

The rules are given below, grouped in the 13 windows used to structure them. Text between asterisks are comments. Window titles are given in italics. Figure D4 gives an indication of the structure of the knowledge by showing which windows have rules called by rules in other windows.

*Eligibility for Child Benefit*

```
/* Date is the Monday of the week about which the claimant is making the enquiry */

entitledTo(Date, childBenefit, Claimant, Child, Amount) :- eligibleFor(Date, childBenefit,
      Claimant, Child), amount(Date, childBenefit, Claimant, Child, Amount).
eligibleFor(Date, childBenefit, Claimant, Child) :-      qualifyingChild(Date, Claimant,
      Child)

amount(Date, childBenefit, Claimant, Child, Amount) :-     childBenefitAmount(Date,
      Amount)
```

*Benefit rates*

```
childBenefitAmount(Date, Amount) :-     before(Date, 25/11/85), Amount is 7.00.
childBenefitAmount(Date, Amount) .-     between(Date, 25/11/85, 25/11/86), Amount
      is 7.10.
childBenefitAmount(Date, unknown) :-    after(Date, 25/11/86).
```

*Qualifying Child*

```
qualifyingChild(Date, Claimant, Child) :-     residenceConditionOfChild(Date, Child,
      Claimant), careAndCharge(Date, Child, Claimant), ageAndEducationalStatus(Date,
      Child).

residenceConditionOfChild(Date, Child, Claimant) :-     residentInGB(Date, Child).
residenceConditionOfChild(Date, Child, Claimant) :-     livingInGB(Date, Child),
      qualifyingClaimant(Date, Child, Claimant), isKindOfParent(Date, Child, Claimant).

careAndCharge(Date, Child, Claimant) :-     livingWith(Date, Child, Claimant).
careAndCharge(Date, Child, Claimant) :-     not livingWith(Date, Child, Claimant),
      inTheCareOf(Date, Child, Claimant).

ageAndEducationalStatus(Date, Child) :-  ageGroup(Date, Child, under16).
ageAndEducationalStatus(Date, Child) :-  ageGroup(Date, Child, '16To19'),
      inSchooling(Date, Child).
ageAndEducationalStatus(Date, Child) :-  ageGroup(Date, Child, over19) /* mentally
      handicapped etc */, fail.

inSchooling(Date, Child) :-  inFullTimeEducation(Date, Child),
      atEducationalInstitution(Date, Child).
inSchooling(Date, Child) :-  leftSchoolThisTerm(Date, Child),
      beforenextTermAndNoJob(Date, Child).

beforenextTermAndNoJob(Date, Child) :-  beforeNextTerm(Date), not
      inFTJobOrYTS(Date, Child)
```

```
atEducationalInstitution(Date, Child) .-   atSchool(Date, Child).
atEducationalInstitution(Date, Child) :-   atOtherEducInst(Date, Child).
atOtherEducInst(Date, Child) .-   educationAtSchoolLevel(Date, Child), not
      financialSupportEmployerOrMSC(Date, Child).

inFTJobOrYTS(Date, Child) :-   inFTJob(Date, Child).
inFTJobOrYTS(Date, Child) :-   onYTS(Date, Child).
inFTJobOrYTS(Date, Child) :-   onEmployerPaidEduc(Date, Child)
```

*Residence in Great Britain*

```
residentInGB(Date, Person) :-   livingInGB(Date, Person), daysInGBInLastYear(Date,
      Person, Days), Days >= 182.

livingInGB(Date, Person) :-   permanentAddressInGB(Date, Person).
```

*Having prior claim*

```
topPriority(Date, Claimant).
```

*Responsibility*

```
inTheCareOf(Date, Child, Person).
```

*Time Rules*
(These had been written previously and not all are relevant)

```
/* rules for finding out whether a specific time is during a period. Rules are of form:
      period(Date, Person). They succeed only if the Date is in the period.
Person is married if Date is after their wedding to Spouse, and before spouse's death if
      spouse is dead -   note no divorce in this world! */

married(Date, Person, Spouse) .-   wedding(Person, Spouse, WeddingDate), died(Spouse,
      DeathDate), between(Date, WeddingDate, DeathDate).
married(Date, Person, Spouse) :-   wedding(Person,Spouse,WeddingDate), not
      died(Spouse, DeathDate), after(Date, WeddingDate).



/* utilities */

intervalInWeeks(EndDate, StartDate, Period) :-   days(StartDate, Start), days(EndDate,
      End), Period is (End -   Start) // 7.

addYears(D/M/Y, Years, D/M/YY) :-   YY is Y + Years.

between(Date, StartDate, EndDate) :-   days(Date, Days), days(StartDate, Start),
      days(EndDate, End), Days >= Start, Days =< End
```

```
/* succeeds if Date is after AfterDate */

after(Date, AfterDate) :-  days(Date, Days), days(AfterDate, After), Days > After.

/* succeeds if Date is on or after AfterDate */

onOrAfter(Date, AfterDate) :-  days(Date, Days), days(AfterDate, After), Days >= After.

/* succeeds if Date is before BeforeDate */

before(Date, BeforeDate) :-  days(Date, Days), days(BeforeDate, Before), Days < Before.
```

*Qualifying claimant*

```
qualifyingClaimant(Date, Child, Claimant) :-  residentInGB(Date, Claimant),
    hasCareOf(Date, Child, Claimant), topPriority(Date, Claimant, Child), isAdult(Date,
    Claimant).
```

*Family relationships*

```
isKindOfParent(Date, Child, Parent) :-  parent(Child, Parent).
isKindOfParent(Date, Child, Parent) :-  stepParent(Date, Child, Parent).
isKindOfParent(Date, Child, Parent) :-  adoptedParent(Date, Child, Parent)
```

*Personal rules*

```
livingWith(Date, Person1, Person2) :-  permanentAddress(Date, Person1, Address),
    permanentAddress(Date, Person2, Address), Person1 \= Person2.

ageGroup(Date, Person, under16) :-  dateOfBirth(Person, DoB), addYears(DoB, 16,
    SixteenthBday), before(Date, SixteenthBday).
ageGroup(Date, Person, '16To19') :-  dateOfBirth(Person, DoB), addYears(DoB, 16,
    SixteenthBday), after(Date, SixteenthBday), addYears(DoB, 19, NineteenthBday),
    before(Date, NineteenthBday).
ageGroup(Date, Person, over19) :-  dateOfBirth(Person, DoB), addYears(DoB, 19,
    NineteenthBday), after(Date, NineteenthBday).

inFullTimeEducation(Date, Child) :-  not leftFullTimeEducation(DateLeft, Child).
inFullTimeEducation(Date, Child) :-  leftFullTimeEducation(DateLeft, Child), after(Date,
    DateLeft).

beforeNextTerm(Date, Child) :-  leftSchool(DateLeft, Child),
    startDateOfNextTerm(DateLeft, StartDate), before(Date, StartDate).

atSchool(Date, Child) :-  startedSchool(DateStarted, Child), not leftSchool(DateLeft,
    Child).
atSchool(Date, Child) :-  startedSchool(DateStarted, Child), leftSchool(DateLeft, Child),
    between(Date, DateStarted, DateLeft).
```

```
inFTJob(Date, Person) :-  startedFTJob(JobStartDate, Job, Person),
      endedFTJob(JobEndDate, Job, Person), between(Date, JobStartDate, JobEndDate).
inFTJob(Date, Person) :-  startedFTJob(JobStartDate, Job, Person), not
      endedFTJob(JobEndDate, Job, Person), after(Date, JobStartDate).

inYTS(Date, Person) :-  startedYTS(YTSStartDate, Person), endedYTS(YTSEndDate,
      Person), between(Date, YTSStartDate, YTSEndDate).
inYTS(Date, Person) :-  startedYTS(YTSStartDate, Person), not endedYTS(YTSEndDate,
      Person), after(Date, YTSStartDate).

inEmplPaidEduc(Date, Person) :-  startedEmplPaidEduc(StartDate, Course, Person),
      endedEmplPaidEduc(EndDate, Course, Person), between(Date, StartDate, EndDate).
inEmplPaidEduc(Date, Person) :-  startedEmplPaidEduc(StartDate, Course, Person), not
      endedEmplPaidEduc(EndDate, Course, Person), after(Date, StartDate).

permanentAddress(Date, Person, Address) :-  address(Person, Address, From, To),
      between(Date, From, To).
permanentAddress(Date, Person, Address) :-  address(Person, Address, From),
      after(Date, From).
```

*Personal facts* (This is test 'data')

```
dateOfBirth(betsy, 10/9/73).

address(betsy, '19 Fir Lane, Glasgow', 10/9/73, 1/3/86).
address(betsy, '1 Acacia Avenue, ToyTown, TY1 5WN', 1/3/86).

daysInGBInLastYear(Date, Person, Days) :-  bAsk(['How many days has', Person, 'been in
      GB in the year before?', Date], Days).

permanentAddressInGB(Date, Person) :-  bCheck(['On', Date, ', was ', Person, "'s
      permanent address in GB?']).

leftFullTimeEducation(4/12/86, betsy).

leftSchool(4/12/86, betsy).

/* the following two need more work -  eg a table of term dates (It has subsequently been
      noted from a Child Benefit book that these are not needed. There are 'fixed dates', the
      first Monday in January, the Monday after Easter Monday, and the First Monday in
      September, which define the end of the holidays following the relevant school term)
      */

leftSchoolThisTerm.
startDateOfNextTerm(4/12/86, 19/1/87).

startedCourse.
endedCourse.
```

```
courseSupport.

startedFTJob( 1/1/87, hairdresser1, betsy).
endedFTJob.

startedYTS.
endedYTS.

startedEmplPaidEduc.
endedEmplPaidEduc.
```

*Education*

```
educationAtSchoolLevel( Date, Child) :-   onEducCourse( Date, Child, Course), not
      higherCourseOrTraining( Course).

higherCourseOrTraining( degree).
higherCourseOrTraining( 'HNC').
higherCourseOrTraining( 'HDipTec').
higherCourseOrTraining( profQual).
higherCourseOrTraining( apprenticeship).

onEducCourse( Date, Child, Course) :-   startedCourse( Start, Course, Person),
      endedCourse( End, Course, Person), between( Date, Start, End).
onEducCourse( Date, Child, Course) :-   startedCourse( Start, Course, Person), not
      endedCourse( End, Course, Person), after( Date, Start).

financialSupportEmployerOrMSC( Date, Child) :-   courseSupport( Date, Child, Course,
      employer).
financialSupportEmployerOrMSC( Date, Child) :-   courseSupport( Date, Child, Course,
      msc).
```

*Boring Ask the User*

```
bAsk( Prompt, Answer) :-   prompt_gread( Prompt, Answer).
bCheck( Prompt) :-   yesno( Prompt)
```

Future work

Plans for the future include a systematic comparison of sources used and consideration of the other publicly available knowledge sources, especially the legislation. Only a beginning has been made on this so far.

The potential of the diagrams has not been tested fully. The work so far has concentrated on eligibility rules which have been well within the capabilities of the AND/OR tree and the aspects of the SGN which have been incorporated.

It is an open question whether non-programmer domain experts will 'read' the diagrams differently from the way they were 'read' by non-programmer university colleagues. It would be useful to document the way domain experts, both from within and outside the DHSS, navigate through the intermediate representation and what features make it particularly useful or awkward for them.

Work has begun on the characteristics of the intermediate representation, identifying different kinds of leaf nodes and cross referencing.

## References

Bliss, J., Monk, M. and Ogborn, J.M. Qualitative Datat Analysis for Educational Research. a guide to the use of Systemic networks. London: Croom Helm, 1983.

Butler, Christopher S. (1985) Systemic Linguistics:Theory and Applications, London: Batsford Academic and Educational.

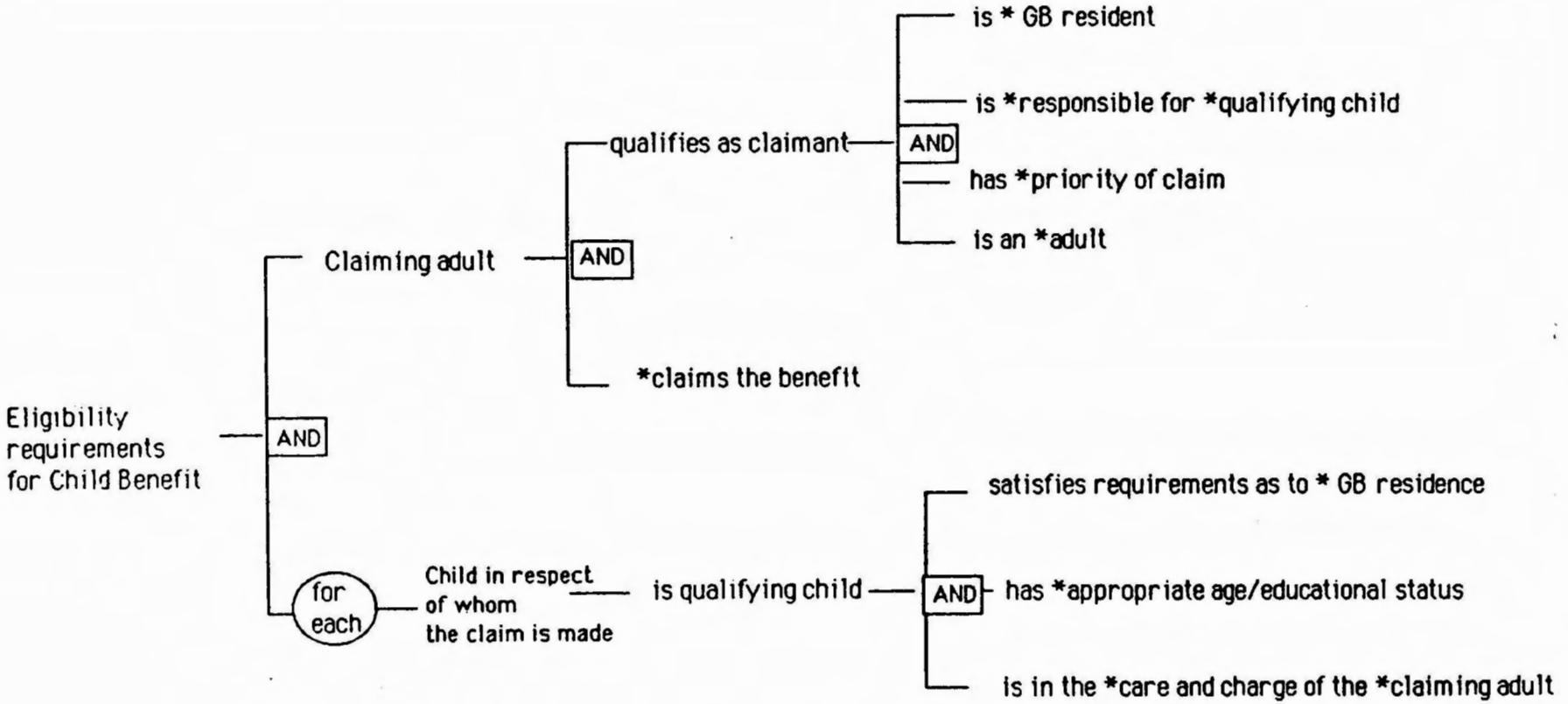
Family Welfare Association (1987) Guide to the Social Services 1987, 75th Edition, London: The Family Welfare Association.

Lynes, Tony (1985) Penguin Guide to Supplementary Benefits, Fifth Edition, Harmondsworth, Penguin.

Matthewman, Jim (1986) Tolley's Social Security and State Benefits 1986. Croydon: Tolley Publishing Company Ltd.

Smith, Roger and Rowland, Mark (1985) rights guide to non-means - tested social security benefits. Eighth Edition 1986, London: Child Poverty Action Group.

# Adapted Systemic Grammar Network Representation 2: Eligibility for Child Benefit




* Indicates there is more detail available elsewhere

for each — This indicates that what follows must be performed for each

Figure D1

FIGURE D2

**The Child** in respect of whom a child benefit claim is made (in week *of the claim)

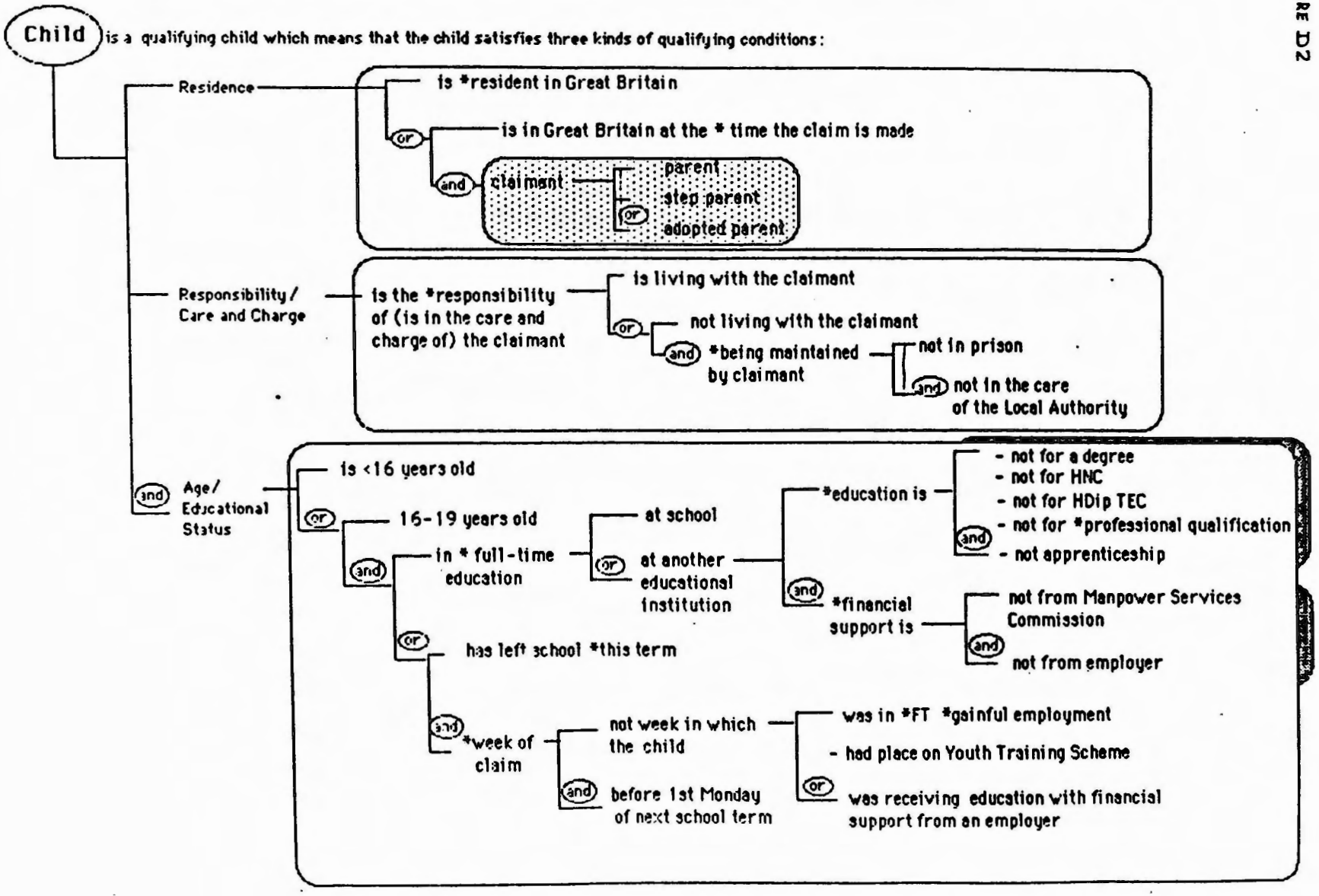
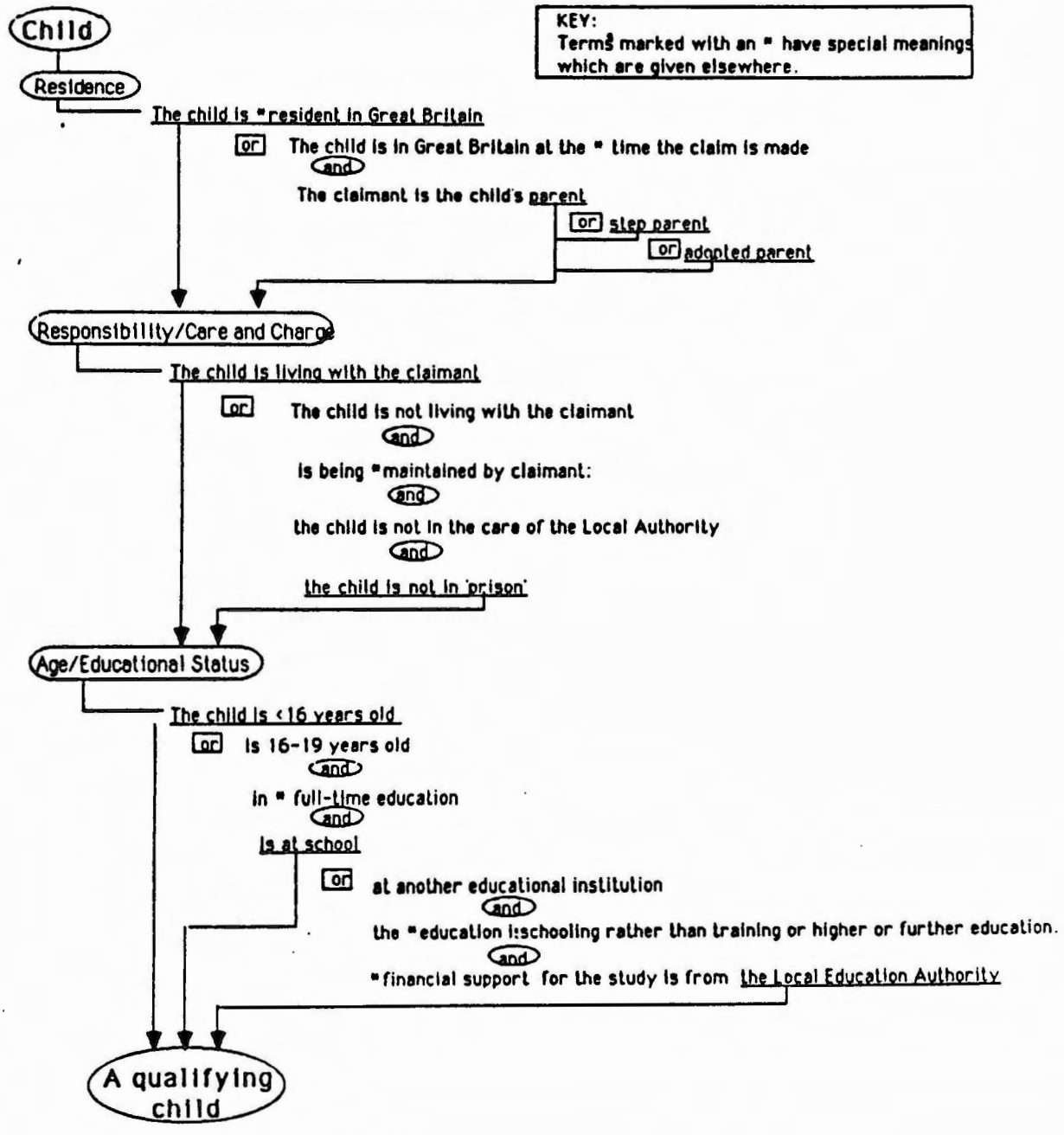


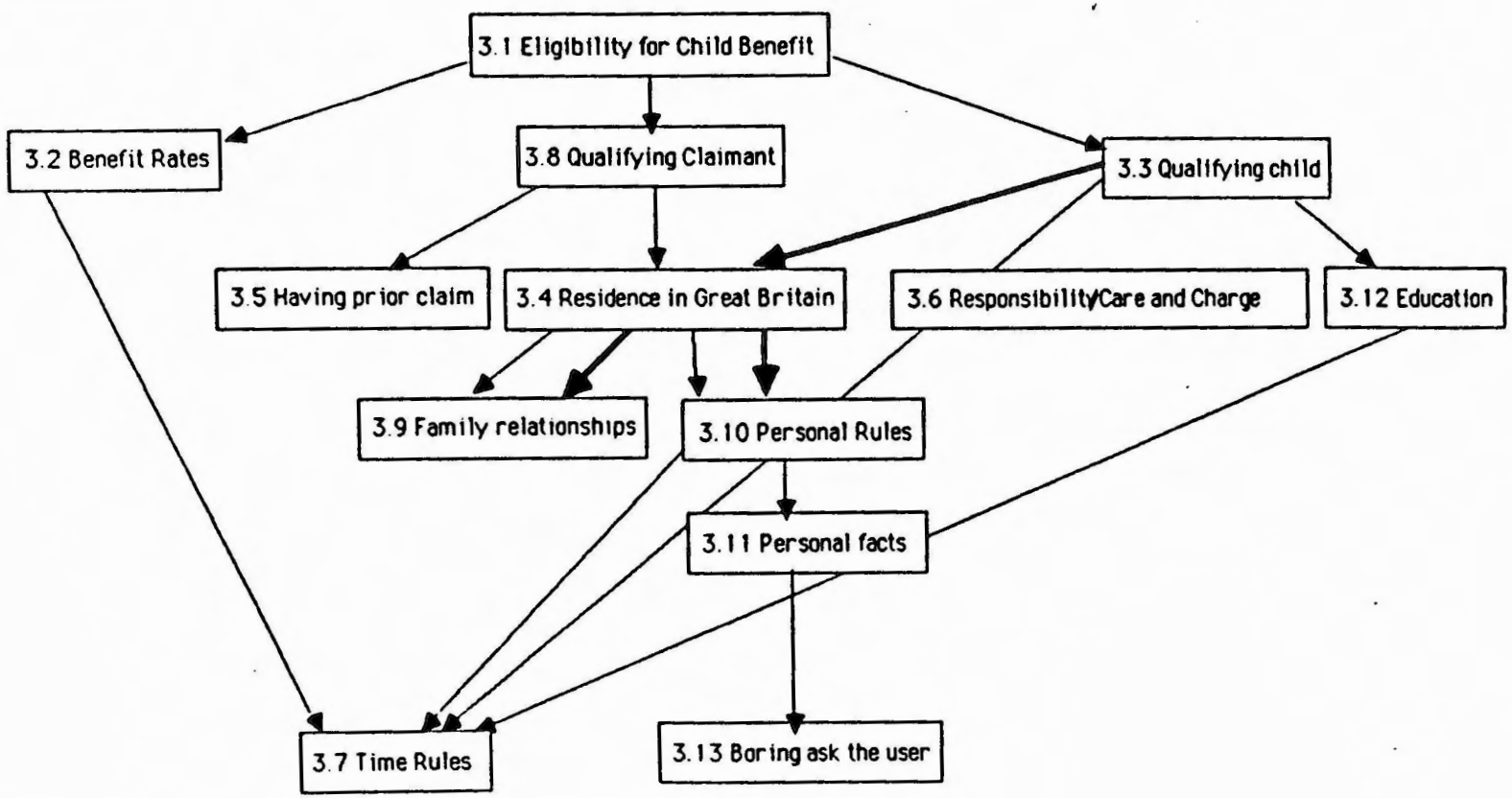
**Figure D3**

**The Child** in respect of whom a child benefit claim is made must (in week of the claim) be a qualifying child. This means that the child satisfies the three kinds of qualifying conditions shown below. If all three conditions are satisfies, someone can claim Child Benefit for that child for the week in question.



## Window Structure of Child Benefit Prolog

Figure D4




The thick lines indicate that the rules in 'Qualifying Child' calls different rules in 'Residence in Great Britain' from those called by 'Qualifying Claimant'

# KNOWLEDGE ACQUISITION AND NUCLEAR SAFETY ADVISORS

Prof Bryan McHugh

15 AUGUST 1987

Institution for Nuclear Technology
Chalmers Tekniska Högskola
S-41296 GÖTEBORG Sweden

Tel: (46) Ø31 721432
Telex: 2369 CHALBIB S

B McHugh

## Introduction

1. This paper has been written to describe the work being carried out at this Institution to establish expert system technology as a useful tool for helping to maintain the safety record of the Swedish nuclear programme and for improving our ability to cope with nuclear emergencies.

2. Two current trends lie behind this work programme. Public opinion in Sweden, and indeed in most Western countries, is vociferous in demanding ever higher standards of safety in the operation of the increasing number of nuclear facilities that exist.

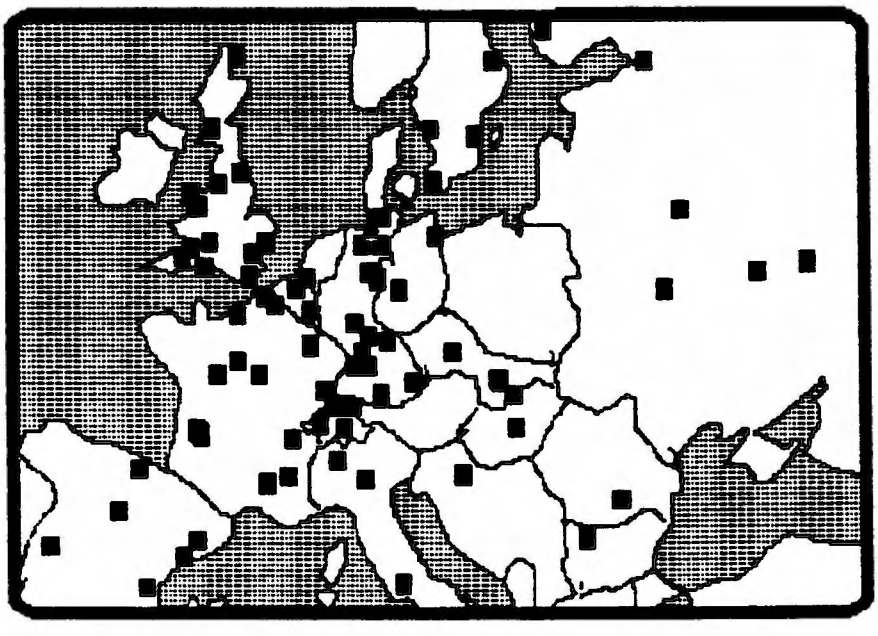


Fig.1 Nuclear Power Sites in Europe.

3. At the same time rapid progress is being made in introducing micro-computer techniques into power plant operations. The thought is perhaps natural then to look to the use of intelligent procedures based on micro-computers to assist plant operators and other authorities in making decisions and in carrying out procedures for both routine and emergency situations in the nuclear field.

4. Two differing areas of activity have been chosen for which prototype "expert" systems have been devised. The first deals with the internal operations at a nuclear plant which are carried out in accordance with certain overiding safety-related rules. These rules are defined by the national safety authorities and must be followed correctly at all times. A rulebook interpreter can be constructed that uses information of a relatively static nature to help identify the correct procedures for any given operational situation.

5. The second system that has been set up concerns the situation external to a nuclear station during an emergency. Here the intent is to provide third parties with advice about potential radiation hazards arising from damage to any given nuclear power plant in Europe. The information base used for this system is (or should be) dynamic since the status of the nuclear plant concerned changes continuously to reflect power system requirements.

6. The two systems described here have been executed on MS-DOS machines and taken to the stage of being usable products for demonstrating their potential capabilities. Further development hinges on actual use, where the updating of the knowledge bases used will be an issue of prime concern.

## Background

7. The first programme, the rulebook interpreter, has been constructed for the Ringhals-2 plant that is situated on the West coast of Sweden roughly 50 km from the city of Gothenburg. Named PROODOS it has been given the task of presenting the correct passage(s) in the current version of the Technical Safety Specifications for that plant to suit any given operational situation. Actual implementation is on the Ericsson PC machines that have been installed in large numbers at the plant site for general use by plant personnel.

8. A fundamental decision was taken in the design of PROODOS to ensure that the system would not **supplant** the operator in any decision process. Thus the information that is presented to the system comes exclusively from the operator. The system is deliberately isolated from the information and data channels already established on site to operate the plant and to supervise plant status. Thus the system will give "conclusions" and advice based solely on what the operator presents to it. Hopefully this will sustain confidence in the operation of the system by obviating responses based on hidden (automatic) or possibly suspect sources of data.

9. A further design decision was to write the system programme using a Prolog dialect suited to the PC in question. The decision processes involved in selecting the correct passages in the rulebook are rather simple; in principle any programming language that can handle a database of text and that can run a nice screen presentation would be suitable. Turbo-Prolog was selected since it gave us an opportunity to combine systems development with a pedagogical exercise in programming with this language.

10. The second system to be developed is an entirely different proposition. Here the problem is to provide an anxious user with an estimate of the potential irradiation that can arise as a result of some "accident" that has reportedly happened at some nuclear plant "out there" somewhere.

11. The essence of the problem is the handling of uncertain information. As was demonstrated acutely during the Tjernobyl incident, it can be very difficult to obtain hard facts about the extent of any damage, the amount of release, if any, and what the chances of local effects are.

12. A system that can be fed with such information as is available, or conjectured, and which can supply a non-expert with an understandable and concrete prediction of possible risk is felt to be of potential value to many individuals and institutions.

13. A programme called Nuclear Incident Threat Analyser (NITA) has been designed for use in a PC-environment using a commercial shell - SAVOIR - which provides for the handling of incomplete information, and which can estimate upper and lower limits or probabilities of numerical expectations of the consequences of chains of events. In this case what facts as to extent of plant damage, and certain meteorological data as can be stated (or guessed) are combined with a knowledge base that gives information on plant location, type, size, status, etc. together with certain models of the spread and effects of radioactive isotopes in the biosphere.

## PROODOS
## A rulebook interpreter

### General

14. Turbo-Prolog provides for the consultation of databases as part of the built-in predicate library. It is thus easy in principle to search through a database for a desired conjunction of input data, and come up with the required information.

15. One constraint that pertains to this version of Prolog is that despite it being possible to open and close any desired file of data, only **one** record format is possible in the programme module. This means that only one universal record

format can be used for the information that is to form the knowledge base(s) on which programme functioning will be founded.

16. In action the programme assembles information concerning an given operational phenomena and identifies the situation as belonging to either a specific event or class of events. The text - taken from the Technical Safety Specifications - that is applicable to the event(s) identified is then presented to the operator.

17. It is thus necessary to establish two interactions with the knowledge base available to the programme. Firstly a number of descriptors of the situation in question are matched to an event description. For example:

> If the high neutron flux measurement channel shows a response time greater than 0.7 secs when in power operation , then
>
> a matching is made to an event coded L47 via descriptors that identify the status of the plant as being in **power operation**, the portion of the plant affected as being **system 352** in the **plant control and instrumentation equipment** portion of the plant, and the event in question being a **slow response time**.

Secondly the knowledge base is searched for a text that is associated with the event code **L47**, and that text - which is taken straight out of the safety instructions - is then set up on the screen as advice to the user.

18. As a support and verification of the advice issued by the programme, it was judged necessary to enter the entire text of the regulations and requirements included in the rulebook into a separate programme module - also written in Prolog. This text was then equipped with indexes and content lists and given a search facility so that the relevant and supporting text can be referred to. This was purely to backup the function of the main programme and to sustain user confidence in the selection of advice made by the main programme.

19. It should thus be noted that the programme offers two ways of finding the relevant portions of advice text. a) A simple search through the rulebook using

contents and indices in a word-processor like manner. b) Identification of the event that has occasioned interest via a dialogue with the user whereby the equipment giving rise to concern is given an identity, and the possible "symptoms" (as described in the rulebook) pinpointed. Text associated with the event is then found.

20. This latter way of finding correct event descriptions contains the essence of the artificial intelligence approach. Further developments of the system will focus on this aspect of the programme.

## Knowledge base for PROODOS

21. Seven different files are maintained to comprise the knowledge base used. Updating can be done of these files in any suitable word-processer (PC-Write is excellent) without disturbing the executable programme code. As mentioned above a common format has to be employed on all files and records..

22. The files are grouped into three different types. The first file is used to define the status of the plant (shutdown, refuelling, power operation etc) and given an operating code number. The second group of four files contain information on the components involved. This is done by using a three-rank hierarchy (three digit identification number for the equipment system involved) plus an individual item identification. These identifications are not taken beyond the level used in the rulebook. The third group contains classifications of the "behaviour" of the equipment that calls for action on the part of the operator. The final file contains the text that is associated with the behaviour.

item("plant_status","No$^5$","reference","","","","","","") status.dat

item("comp$^1$ ","No$^1$"," reference","","","","","","","") comp1.dat
item("No$_1$ ","comp$^2$ "," No$^2$","ref ","","","","","") comp2.dat
item(" No$^1$","No$^2$ "," comp$^3$","No$^3$ "," reference","","","","") comp3.dat
item("No$^1$ ","No$^2$ ","No$^3$ "," comp$^4$"," No$^4$"," reference","","","") comp4.dat

item("No$^5$ ","No$^1$ ","No$^2$ "," No$^3$","No$^4$ ","indication "," behaviour"," reference") behave.dat
item("indication "," action","reference","","","","","","") action.dat

Fig.2 Knowledge Base Files

23. The required text in held in "action", and is selected by finding the proper "behaviour". This in turn is found from the correct combination of No$^5$, No$^1$, No$^2$, No$^3$, and No$^4$ as given by the operator.

24. These files need only be updated at infrequent intervals since plant equipment is not often subjected to changes in procedures or in observed behaviour.

## NITA
## Threat analysis programme

### General

25. This programme is written using SAVOIR. This provides in essence a production-rule type environment for deriving solutions to problems posed by the user. Having only a very primitive ability to communicate with external information, it is necessary to include all information in the source code. This means that although the source code can be edited using any suitable word-processer a subsequent compilation calls for the SAVOIR programme package to be available.

B McHugh

26. Use of NITA is by selecting a nucear station for study by selecting from country lists as an interactive excercise on the computer screen. This is followed by entering estimates - guesses or hard facts - about the extent of possible damage to the nuclear plant - containment and primary circuits and core components. (In the absence of facts the programme supplies ranges of possibilities.)

27. Information is then presented to the user concerning distance, direction etc from the plant site to the users location (in this case Gothenburg, Sweden). Information on the weather situation and diurnal cycles during possible transit to Gothenburg are then entered (or default ranges used).

28. Using built in correlations it is then possible to present possible airbourne contamination levels, rates of deposition and uptake in milk etc. This information is then presented in the form of possible action-levels at which protective measures can be called for.

29. A considerable amount of textual information can be presented on demand as tutorial information on radiation, its effects and advisable countermeasures.

30, NITA also includes in the built-in knowledge base local information concerning civil defence resources, contacts and procedures, thus providing a complete overview of the local facilities and how to make use of them when required.

## Knowledge base for NITA

31. Information about individual reactor plant are inserted into the SAVOIR source code as quasi-English language statements:

```
IF reactor= Ringhals_2 THEN distance=52 AND power=800

IF reactor= Ringhals_2 THEN  type=pwr AND release=0.3
```

thus setting variables such as distance, power, type and release to values for the actual reactor plant set up for investigation.

32. SAVOIR permits the use of Bayesian probability calculations where suitable for the combination of factors subject to uncertainty, and maintains both upper and lower bounds on all variables. This allows for handling uncertainty in the knowledge base itself where actual facts are in dispute.

## CONCLUSION

33. The knowledge bases involved in the two programmes dealt with here have no large volume. In the case of PROODOS a matter of 60 individual events, or classes of event were extracted from the rulebook ( roughly 150 pages of prime information). For NITA 89 individual reactor plant were identified for the purposes of the programme.

34. Maintaining the information does not present much of a problem, and for the case of the Prolog programme it would be a minor matter to produce an editing procedure with such built-in checks that the unschooled user could maintain the rulebook interpreter in working order.

35. In the case of NITA the quality of the information required from the knowledge base to produce reliable results is such that considerable expert knowledge is needed. It is thus quite acceptable that the insertion of extra information can only be done by the programmer with access to the SAVOIR compilation package.

## ACKNOWLEDGEMENTS

Thanks are due to the Swedish Nuclear Inspectorate for their support of the development of the PROODOS programme. Thanks are also due to Dr George Fileas of my institution who did the Prolog programming.

NITA was developed by me without any external support.

# EXPERIENCE OF KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS IN CONSTRUCTION

E G Trimble, Professor of Construction Management, Department of Civil Engineering, Loughborough University of Technology.

C N Cooper, Research Assistant, Department of Civil Engineering, Loughborough University of Technology.

## INTRODUCTION

The authors are currently engaged in research into methods of knowledge acquisition for expert systems. Although their original focus was on construction industry applications the findings so far suggest that the choice of industry does not of itself affect the knowledge acquisition method selected. Rather it is the situation that determines the appropriate method. By situation we mean such factors as:

- Available time of the domain expert
- Whether he holds his expertise in explicit or intuitive form
- Whether he is motivated to help the process or hinder it.

The authors are associated with two research projects into knowledge acquisition methods. The first involves the development of a number of expert systems in conjunction with industrial clients as summarized below:

CONPLANT - selection of materials handling equipment for multi-storey construction sites.

BREDAMP - diagnosis of the cause of dampness in buildings.

CRANES - selection of tower cranes for multi-storey construction sites (incorporating a graphic interface).

BIDDER - decision of a design and construct contractor on whether to bid for a project.

NETWORK - diagnosis of faults in a national computer network.

MATSEL - material selection for boiler pressure parts.

The first five systems utilise the SAVOIR commercial shell program and run on an IBM PC XT. The last named uses the IBM shell ESE and runs on an IBM mainframe computer. Different client organizations and experts have been involved in each case and this has enabled the authors to gain a broad spectrum of experience of knowledge acquisition.

The second area of research is being undertaken by Mr Alan Bryman and Dr Joe Cullen of the Department of Social Sciences at Loughborough University in association with Professor Trimble. The aim of this work is to look at the experiences of industry in developing in-house expert systems. The work involves questionnaire surveys and interviews with staff from a broad range of companies.

Some of the author's experience of knowledge acquisition will now be described. Much of this material was recently presented at a conference in Washington D.C. but some adjustments have been made to reflect recent findings.

## SITUATIONS THAT AFFECT THE ACQUISITION PROCESS

It is clear that the nature of the situation within which the knowledge is acquired will have a major influence on the knowledge acquisition methods to be selected. The categories so far identified are:

1. The knowledge is held in a largely intuitive undefined format.
2. As category I but some closely similar domains have been examined previously.
3. Cases can be defined that reflect a body of decision-making within the domain.
4. There is published material about the domain.
5. The domain expert has sufficient knowledge about expert systems to enable him to define the knowledge (or at least to play a significant role in its definition).

Superimposed on this list of categories are other dimensions such as:

- o The "depth" of knowledge to be represented, i.e. does it represent fundamental knowledge such as that relating to molecular structure or "heuristic" knowledge which includes a substantial amount of personal opinion.
- o The attitude of individual experts to the system.
- o The extent to which a consensus among experts can be found.

The foregoing categories are now elaborated.

Intuitive knowledge

Some knowledge engineers favour a method which requires the development of a prototype system based very often on the prior knowledge of the knowledge engineer. The prototype is demonstrated to the domain expert who suggests modifications and amplification. The changes are made and the revised system demonstrated again. The iterations of this process continue until the domain expert is satisfied that the model is acceptable. If a good initial model is produced this method can be very productive. However it can have the effect of prejudicing the responses of the expert and thus diverting him from some of the subtle, more intuitive knowledge, that might be of crucial importance in the operation of the system.

An alternative is to start with a blank sheet of paper and ask the domain expert to tell you what he knows. A fairly extensive set of knowledge is then assembled before the initial system is coded. This approach is fundamentally better but its success is critically dependent on the time that the domain expert can devote to the process.

Intuitive knowledge with precedents

Where systems have been produced for very similar domains it may be safe to introduce a short-cut in the form of structured interviews based on the content of the previous systems. The danger of prejudicing

responses must always be borne in mind.

Defined cases

There are several computer programs that will induce rules from sets of cases. Of these EXPERT-EASE is probably the simplest and best known. At first sight this approach has much to recommend it. However, extensive trials of the early programs have revealed some disconcerting problems. One of these is that the natural sequence of questioning that is inherent in a domain is not respected. For example a pair of questions might be:

o Is the pipe a drain?

o Have you performed a drain-test?

If the sequence of these questions is reversed as it may be in rule-induction the confidence of the user will quickly evaporate. Another problem is that, like regression analysis, rule-induction works on cases irrespective of any causal connections.

It is not, of course, imperative to use the rule-induction program. Manual examination of sets of cases will often indicate relationships that can be coded on an ad hoc basis. The expert will frequently find it easier to recall his knowledge through the recounting of case histories than through other forms of interview.

Published material

There is a lot of interest in the use of expert systems to guide users in the interpretation of regulations and codes of practice. Clearly, in this situation, there should be no problem of human interaction as the views of the human experts should be fully recorded in the published text. As an aside it should be noted that attempts to "computerize" regulations were made before the recent surge of interest in expert systems. These attempts often revealed inconsistency and vagueness which made full "computerization" difficult. Some investigators have suggested that this

should be anticipated since differences between the views of the members of the drafting committee eventually have to be resolved by compromise.

Coding by the expert

When the domain expert is also a reasonably competent user of computers it may be possible for him to produce his own expert system without the use of an intermediary. This approach is only possible where the expert exhibits a high degree of self-knowledge and is likely to be unsuccessful where the knowledge is largely intuitive. The expert may or may not be inclined to use an expert system shell to assist him (and constrain him) in his efforts.

SELECTING THE METHOD

The previous section identifies the following methods of knowledge acquisition:

- o Unstructured interviews
- o Structured interviews
- o Case histories
- o Prototype system evolved iteratively
- o Rule induction

To this list should be added:

- o Observational

In the observational method the knowledge engineer observes the domain expert as he performs tasks which require him to draw on his expertise.

- o Paper models

We have recently focussed our attention on the use of paper models after seeing the successful use of this technique by the ARIES Alvey Community Club. The knowledge engineer creates a document detailing the rules elicited and develops this as knowledge acquisition progresses. The document records the status of each rule - i.e. finalized, tentative, needs

review etc and also records the source from which each rule was obtained. A natural language format is adopted so that the paper model can in many cases be reviewed by the domain experts. We have found that this review process can itself stimulate further knowledge acquisition.

o o o

Our studies started with the view that we should be able to isolate situations (or combinations of factors) that would point to the selection of a single method. Our experience has not borne out this view and for example in one of our applications five different methods were used at different stages. Thus our current advice is to acquire a feel for the alternative methods and to use them flexibly as the position unfolds. The following comments augment those in the preceding section:

Unstructured interviewing has the great merit of not prejudicing the responses of the domain expert. Thus less obvious points emerge that can be very important. The method however is time-consuming and requires patience on the part of the expert.

Structured or focussed interviewing achieves results quickly and is appropriate when the knowledge engineer is fairly confident of his understanding of the domain. This understanding may result from prior knowledge or from the results of earlier acquisition methods. Interviews can be broadly focussed or narrowly focussed. During a broadly focussed interview a knowledge engineer might pose questions such as "describe the parts of a typical boiler system" whereas during a narrowly focussed interview "are there any welding problems associated with carbon steel?" would be more typical.

Prototyping has much to recommend it particularly as each interaction can provide cues to prompt the expert in his thoughts about his intuitive knowledge. As with structured interviewing there is a danger that less obvious points may get overlooked.

The authors have very limited experience of the observational method. However it must be a beneficial approach in providing at least the initial

evidence that the knowledge engineer will require in structuring the problem. Furthermore observation can be used to check that the expert performs his tasks in the way he claims to use during interview sessions, i.e. during interview sessions he may be telling the knowledge engineer how he thinks the job ought to be done rather than how it is done in practice.

Rule induction appears to be satisfactory for simple well-defined applications. However for applications even of quite modest levels of complexity we have found that rules prepared by induction are unsatisfactory for direct incorporation in the system. Despite these shortcomings we have found that attempts to apply rule induction to limited modules of a total application can force the expert into considering factors that are not revealed by other methods. This must improve the validity of the knowledge base even if the induced rules are themselves discarded.

## SOME HUMAN FACTORS

Knowledge acquisition for expert systems is a human process and several of the human aspects have already been mentioned. The purpose of this section is to itemize the human problems that arise so that readers can be aware of them. This is not to suggest that we can yet offer solutions; the process is likely to remain largely ad hoc for some time. Before proceeding it is worth reminding ourselves that the process of knowledge acquisition is the transfer and transformation of expertise from some source (usually human) to a computer program.

### Resistance

The domain expert may fear that, by giving up his knowledge, he will weaken his position within his organization. Unless some incentive can be engineered such an expert is unlikely to provide the basis for a useful system. Organizational resistance may also arise and has been observed in

the Community Clubs established in Britain by the Alvey Directorate. For example one club member may provide an expert but then realize that commercially valuable skills could be transmitted via the system to a competitor. It should be noted, on the other hand, that positive motivation may be encountered when an expert is bored with providing personal advice in one subject and would welcome the chance to have this process automated.

Accessibility and prejudicing responses

An expert may have the necessary knowledge and motivation but may have other duties that prevent his spending an adequate amount of time with the knowledge engineer. We have already mentioned the dangers of prejudicing responses by over-structuring interviews and by offering detailed prototypes. However, the methods that prejudice responses are usually quicker so some compromise will often be necessary.

Cues and examples

Experts are often better at doing things than explaining what they are doing and why. So one method of obtaining knowledge is to watch the expert at work and then ask why he did what he did. The problem is that the expert often cannot recall from his subconscious the rules and relationships that have become intuitive. A method that also deals with this problem is to generate artificial examples as cues and to ask the expert what he would do in these circumstances. Our experience in obtaining the Bayes factors for BREDAMP is an illustration of this method (see below).

Rapport and roles

Clearly the knowledge acquisition process will proceed more smoothly and effectively when rapport is established between the knowledge engineer and the domain expert. As a corollary to this, it is usually better

to separate the tasks of knowledge acquisition from those of coding the information for the computer. This enables the knowledge engineer to concentrate on the knowledge as perceived by the expert and on establishing a good human relationship with him.

Much of our work has been undertaken in construction industry domains and we believe that our engineering background has enabled us to develop a rapport with the domain experts more easily than would have been the case had we been from an unrelated discipline. However a knowledge engineer in this situation must be careful to avoid distorting the expert's knowledge by introducing his own ideas.

## SOME PRACTICAL NUANCES

### Development environment

So far the great variety of domains that have been attempted and the approaches adopted have made generalized analysis virtually impossible. However, one conclusion is inescapable, namely that there are at least two distinct kinds of situation namely:

- A client has identified his own need for an expert system and engaged an employee or contractor to deliver a system to the client's requirements.
- An enthusiast, typically an academic, has defined an interesing application and has persuaded host organizations to provide relevant knowledge.

The nature of the relationship between the client (or host organisation) and the knowledge engineer is of crucial importance. Where a client has defined his own needs it is likely that the experts will be readily available and that they will be uninhibited by company policy and commercial confidentiality. They may still be inhibited by their own motivations. At one extreme they may be eager to impart the knowledge in order that the expert system will eventually relieve them of routine tasks which have degenerated into boring chores. At the other extreme they may fear that

revealing their knowledge will undermine their security and generate a situation where they become redundant.

Our experience is that where an application is undertaken by an enthusiast the objectives and in particular the intended uses to which the system will be put are often not clearly defined. The experts may also be less readily available and inhibited by commercial considerations. As a result such systems are much less likely to succeed than those which are client inspired. However this type of situation may in some cases have its compensations when the enthusiast himself has substantial domain knowledge.

A possible approach to development

We have found that the approach to knowledge acquisition must be flexible and be specific to the domain under consideration. Our own general approach can be characterised as the following progression:

(i) one or two unstructured sessions

(ii) case histories and broadly focussed interviews

(iii) narrowly focussed interviews

(iv) prototyping and narrowly focussed interviews

In our work the demonstration of a prototype system to the experts has provided a valuable stimulus to further knowledge elicitation. We have found that the point at which the expert first sees this prototype should be selected with care. On the one hand the knowledge engineer may become prematurely over-enthusiastic about the system and as a result demonstrate a piece of code which is trivial, thus losing the confidence of the expert. On the other hand we have encountered an expert who had serious misconceptions about the capabilities of the system being developed, and such misconceptions must be dispelled at an early stage. In the light of experiences such as these our advice is to demonstrate the prototype as soon as it is capable of giving a piece of recognisable and plausible advice.

Following our most recent work we also recommend that a paper model should be developed as a means of documenting the knowledge acquisition process and as a focus for discussion with the domain experts.

Uncertainty

The BREDAMP system generated some useful insights into the problems of uncertain knowledge. This system was commissioned by the Building Research Establishment; its purpose is to diagnose the cause of dampness in buildings. The domain expert was exceptionally cogent and well motivated. However it was necessary to attach probabilities to the goals e.g. to conclude that there was a 9O% likelihood that the cause of the dampness was rising damp. For this the dependencies between variables are calculated using Bayes theorem for which affirmative and negative factors must be established. While we could expect the domain expert to describe the behaviour of dampness phenomena it was impractical to obtain from him estimates of the BAYES factors. To overcome this problem we first obtained from him the key factors relating to each cause (or goal). We then compiled tables in the following form and asked the domain expert to suggest values to replace the question marks:

| Factor | Suggested values | |
|---|---|---|
| Evidence | A stain | A stain |
| Height of stain | 9 inches | l5 inches |
| Age of building | 8 years | 9 years |
| Component wetter inside than out | Yes | Don't know |
| Positive salts test | Don't know | Yes |
| Probability of rising damp | ? | ? |

To derive BAYES factors from these data it was sufficient to use an ad hoc approach i.e. a combination of simultaneous equations and trial and error. A better approach would have been some form of regression analysis

although it can often be difficult to elicit a sufficient number of cases to make this approach possible.

This case illustrates a further point namely confidence limits. At present BREDAMP offers only a set of probabilities for each of the defined causes of dampness. For example:

| | |
|---|---|
| Rising damp | 90% |
| Rain penetration | 27% |
| Others | less than 5% |

The rising damp figure may in fact mean that the probability is in the range 89-91% or it may mean that it is in the range 80 - 100%. A user would react differently if he had these ranges available. With a narrow range he is likely to conclude that he has gone as far as the system will allow and he may then decide to take remedial measures to cure the problem on the assumption that the cause is in fact rising damp. If the wider range (80-100) is shown he will probably undertake additional, quite cheap, tests to improve the reliability of his diagnosis. This extension of the information provided by a system has been mentioned in several contexts, but no actual implementation has so far been identified by the authors.

## CONCLUSIONS

The following are offered as reminders of the key points in this paper:

- o System development should be client led.
- o Knowledge acquisition methods depend much more on situation than domain.
- o Flexibility of approach is essential to the knowledge acquisition process. Factors which will determine this approach include:
  - the form in which the knowledge is available
  - the depth of knowledge (i.e. fundamental or heuristic)
  - the degree of consensus among experts
  - the attitudes of individual experts to the system.

- o It is unlikely that a single method of knowledge acquisition can be adopted for development of an application - rather a number of methods will be required.
- o Cues and examples can help an expert to recall intuitively held knowledge.
- o Using a computer program to induce rules from cases may provide some enlightenment but is unlikely to provide working rules except perhaps for simple systems.
- o Even with a very responsive expert ascertaining Bayes factors is best done by examples.
- o Our experience has been that the prototype system should be demonstrated to the experts as soon as it starts to give plausible and recognisable advice. This dispels misconceptions at an early stage and in general provokes further knowledge elicitation and promotes enthusiasm on the part of the expert.
- o We have recently had good experience using a paper model as an intermediate representation for the knowledge elicited. This helps the knowledge engineer cope with the mass of information gathered, and with the right experts can itself be used as a knowledge elicitation tool.

FURTHER READING

Welbank, M. A review of knowledge acquisition techniques for expert systems. British Telecom research laboratories, 1983.

Gotts, N.M. Knowledge acquisition for medical systems - a review. Artificial Intelligence in Medicine Group, University of Sussex, 1984.

Buchanan, B.G. Some approaches to knowledge acquisition. Dept. of Computer Science, Stanford University, 1985.

Allwood, R.J., Stewart, D.J., Trimble, E.G., Shaw, M.R. and Smith, J. BREDAMP - an expert system to diagnose the causes of dampness in buildings. Dept. of Civil Engineering, Loughborough University of Technology and Building Research Establishment. To be published shortly.

Cooper, C.N. CRANES - a rule-based assistant with graphics for construction planning engineers. To be presented at 3rd Int. Conf. on Civil and Structural Engineering Computing, London, september 1987.

Wijesundera, D.A. and Harris, F.C. The integration of an expert system into the construction planning process. Proc. of 2nd Int. Conf. on Civil and Structural Engineering Computing 1985, Vol 2, pp399-405. (Describes CONPLANT).

# KNOWLEDGE ACQUISITION: A SURVEY AND BRITISH TELECOM EXPERIENCE

Margaret Welbank
British Telecom Advanced Systems Unit
14 Finsbury Circus, London EC2M 7DP

## 1. INTRODUCTION

In 1986 I undertook a literature survey for the Systems Designers Insight Group (Welbank, 1987). The aim of the survey was to identify how the applicability of knowledge acquisition methods for expert systems varied with factors such as the domain, the type of knowledge sought, the experience of the knowledge engineer, etc.

This paper summarises it and compares our experiences in British Telecom. We have been building PC-based expert systems, generally using shells, for several years and are now promulgating the use of the technology to other groups in BT. We find that in this context only a few techniques for knowledge acquisition are really viable given the organisational culture in which we are operating. A number of recent papers on knowledge engineering in practice make similar points (eg Ellman 1987, Inder et al 1987).

The Advanced Systems Unit is also involved in the KEATS project with the Open University (Motta, Eisenstadt et al 1986). This will be briefly described at the end of this paper.

## 2. METHODS FOR THE INTRODUCTORY STAGE

It is widely felt (eg Gammack 1987, Freeman 1985, Breuker and Wielinga 1984) that there is a preliminary stage of knowledge acquisition when it is more important to establish rapport than to collect information. At this stage it is wise to choose a method which the expert feels most comfortable with. This is doubly true in a business setting.

This may rule out the most productive techniques. Very formal or structured methods are often considered to be inappropriate at this stage - for instance card-sorting (Freeman, 1985) or structured interviewing (Breuker and Wielinga, 1984).

However, more natural unstructured methods produce mostly information which cannot be used, especially if the representation chosen is simple and inflexible (production rules, for example).

### 2.1 Some Unstructured Methods and When They are Applicable

A common method of handling the initial session is what may be called the "unstructured interview": asking rather general questions about the field, tolerating digression, tape-recording everything and hoping to extract useful information from the transcript. In my experience, this usually produces a mass of highly detailed, technical information, because the expert is showing off how much he knows, and the knowledge engineer is left

with no idea what the structure of the field is or what are the important principles.

However, the method will at least allow the concepts of the domain to be identified, and its advantages in a business setting make it practically the only method used (Johnson, 1987). It is similar to a business meeting, a situation which is familiar to both the knowledge engineer and the expert. Therefore the knowledge engineer can use it with less fear of alienating the expert, and with no need for specialist training - a very important consideration organisationally where staff turnover is high. We use it in Telecom for these reasons.

Observation of the expert is recommended when the experts are defensive about their knowledge (Shpilberg et al, 1986). It is only advised for experienced knowledge engineers (Frieling et al, 1985): inexperienced knowledge engineers cannot interpret what they see so well.

Participant observation (ie the knowledge engineer becomes an apprentice) is necessary to acquire tacit knowledge - knowledge which is acquired from experience and never overtly recognised, like how to use a screwdriver (Collins et al, 1985). It may therefore be necessary in skill domains if either the users or the knowledge engineer do not have a background in the domain.

The expert may be asked to prepare a short lecture on his field (Gammack, 1987). This works well if the expert is comfortable with it. We have sometimes had experts prepare a very structured account of their domain, and this has been enormously efficient, but we have always left it to the expert to suggest it.

### 2.2 Use of Structured Methods

Structured methods can definitely sometimes be used at this stage. A striking example is BP's use of ID3 induction using the Expert-Ease package (Slocombe et al, 1986). Boose (1984) reports that automated repertory grid programs produce similar effects. Experts are shown how to use the package and provide possible conclusions and lists of parameters. They are impressed to see this information transformed into rules in front of their eyes. A simple classification expert system can be built this way in very little time. Slocombe et al report that experts prefer this to the poor comprehension of knowledge engineers.

It is not clear under what circumstances it is possible to start with structured methods, and whether a computer package is necessary to do it. At present it would seem to be limited to rather shallow classification problems. It should also be noted that Slocombe et al are thought to be somewhat evangelical by their colleagues.

## 3. ANALYSIS AND STRUCTURING

I find analysis and structuring the knowledge to be the most difficult part of building a knowledge-base. Usually there is a

wealth of information, much of which will be left out, but no obvious structure to help in organising it. Highlighting important features is relatively easy, but takes one not much further.

At this stage a model of the domain knowledge is invaluable. It can be used to guide analysis of transcripts, to understand observation, and structured interviews can be built around it. Experienced knowledge engineers probably have such models in their heads and the knowledge of how to apply them is at the heart of their expertise.

Clancey's model of heuristic classification (Clancey, 1984) is useful for analysing domains which fit its pattern (Frieling et al, 1985). More specific versions have been proposed for medical diagnosis (eg Bennett, 1985) and electronic diagnosis (eg Caviedes & Bourne, 1986). It is to be expected that different tasks will use different structures and these will also vary between domains.

An organisation of configuration knowledge is proposed in (Marcus et al, 1985), but the classification paradigm appears the best understood. More research should be done in this area, and knowledge engineers should consciously apply these models and comment on their usefulness.

## 4. METHODS OF FEEDBACK

### 4.1 Advantages of Feedback

There are many advantages to showing the knowledge back to the expert in a different representation:
- the expert is interested.
- he may be more inclined to believe the project is possible.
- having articulated the knowledge once, he has it better structured in his mind and can state it more clearly a second time.
- he can easily see (and correct) any misconceptions the knowledge engineer has.
- the exercise of representing the knowledge has helped the knowledge engineer to understand it.

### 4.2 Rapid Prototyping

Rapid prototyping is the most basic method. It is the best way of convincing sceptical experts or their management that the system can be built. Experts are often surprised and impressed to see their knowledge embodied in an expert system, and can hardly believe that it was all derived form their own statements.

However, the knowledge cannot be examined at a glance, and the representation chosen may be too simple because the knowledge engineer has not understood the true complexity of the domain. Therefore, it is sometimes argued (De Greef & Breuker, 1985; Regan, 1986) that some other representation should be used for immediate feedback. Even so, there should be a prototype within

three months at the latest (Gaines, 1986).

In Telecom, where expert systems technology is largely unfamiliar, we believe in using the prototype as the feedback vehicle because it wins the committment of the expert and shows him what an expert system really is. There are often rather major misconceptions and suspicions which no amount of explanation can really dispel. For the same reason both users and the managers in charge of the project must get a chance to view the emerging system at a very early stage.

### 4.3 Intermediate Representations

A large number of other representation methods have been recommended. The simplest in concept is the paper knowledge base which is the knowledge written out in unambiguous English in suitable chunks for encoding into a system. This can be understood and vetted by the expert and is also a useful deliverable to management. It is part of Teknowledge's methodology. (Frieling et al, 1985). Paper knowledge bases in our experience are particularly useful for explaining the domain to people who have to take over a project (naturally an important factor in an organisational setting), but are no substitute for a face-to-face meeting for gathering the expert's comments.

Graphical representations are often said to be better than verbal ones (although it should be noted that one must be graphically literate to interpret them). Semantic nets, inference nets, systemic grammar networks and decision trees are all quite widely used. But the most positive recommendations are for decision trees (eg Hannan and Politakis, 1985; Alvey et al, 1986). Experts often spontaneously use them when explaining a point and find them easy to understand. They are particularly appropriate for technical diagnosis, where the knowledge is often based on procedures rather than principles. They show up gaps in the knowledge base very clearly and provide information about the control of the consultation as well as domain knowledge. Green and Wood (1984) report that they are more easily understood than inference nets. We use them largely to help us in structuring and designing the knowledge-base. Experts do not always find them very clear and they do not generate so much feedback as showing the prototype.

## 5. METHODS FOR DIFFERENT TYPES OF KNOWLEDGE

One of the briefs for the survey was to compare the knowledge acquisition methods found suitable for different types of knowledge. There has been quite a lot written on this, although the categories identified vary greatly so that it is difficult to compile the knowledge into a common classification.

With this reservation, the following table shows which knowledge acquisition methods have been found appropriate for different types of knowledge. This includes methods which are considered too artificial to be introduced until a good rapport exists with the expert.

| | Facts | Conceptual Structure | Causal knowledge | Rules | Weight of evidence | Procedures | Expert's Strategy | System's Strategy | Context of Rules | Explanation | Justification |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Interview | Y | Y | Y | ? | ? | | N | | ? | | Y |
| Talking through Case Studies | Y | N | Y | Y | N | Y | N | | | Y | |
| Observing Interactions | | | | | | | Y | | | Y | |
| Protocol Analysis | | | | | | Y | Y | | | | |
| Card-sorting | | Y | | | | | | | Key: Y yes N no ? difficult, needs care | | |
| Multidimens'nal Scaling | | Y | | | | | | | | | |
| Repertory grid | | Y | N | Y | Y | N | N | | | | |
| Induction | | | N | Y | | | | | | | |
| Task Analysis | | | | | | | | Y | | | |
| User Interviews | | | | | | | | Y | | | |
| Examining Prototype | | | | Y | Y | | | | Y | Y | |

**Table: Types of Knowledge with Appropriate Acquisition Methods**

The structured interview is a good general purpose method, which is reasonably effective, but it has known deficiencies. It is poor at eliciting the problem-solving strategy used by the expert, the contexts in which rules are true, or weights of evidence for probabilistic reasoning. Also, interestingly enough, it is not as good as other methods for eliciting "if-then" rules. Structured interviews can be quite successful at eliciting poorly remembered knowledge (by asking the same question many times in different ways) and covert knowledge (by asking for examples rather than rules, mimicking an induction package).

Examining transcripts of an expert talking through case studies is rather more laborious, and generates less material ( Burton and Shadbolt 1987). Rules are generated quite easily, but conceptual structure is not, nor is problem-solving strategy, and the method may lead to confabulation in just the same way as the interview (Fox et al, 1985).

Induction and automated repertory grid are very fast and successful for shallow classification problems. They elicit just that knowledge which is necessary to distinguish between the

different conclusions, which makes them very efficient but liable to miss the real richness of the domain.

Protocol analysis is really the only way of discovering the expert's problem-solving strategy, as many people are not aware how they solve problems (this may be domain-specific). However, protocol analysis is not an easy technique for the expert and can only be used when he is convinced nothing else will work (Breuker & Wielinga, 1984). Also, some people (eg Clancey, 1984) feel that trying to emulate the expert's strategy is a mistake, given the limitations of current technology, and the strategy should instead be designed to suit the user's needs.

Conceptual structure is easier to acquire than rules. Concepts are easily identified from unstructured interviews or from s. They can be sorted into groups using card-sorting techniques and the repertory grid can be used to identify the distinguishing features within the groups. These methods are good for eliciting fine distinctions, and they can be used where the knowledge is pictorial and not verbal (Gammack, 1987). Experts can find them irksome and believe they are less efficient than they actually are (Burton and Shadbolt, 1987) which may limit their use in organisations.

Multi-dimensional scaling requires concepts to be presented in pairs and judged for similarity. The process can seem rather meaningless to the expert so it is best avoided unless other methods are not working. But it does produce an efficient classification of the problem space and could be helpful if the expert is not aware of its structure, or to check that the structure he gives is in fact that which he uses (Gammack, 1987).

Examining the prototype is slow, but it is the best way of eliciting information about the context in which the rules should fire (typically the expert finds rules firing when they should not, and then realises that they are only true in certain contexts). It is also good for eliciting explanations and justifications, especially if the user can test the system in the expert's presence.

## 7. ABOUT KEATS

KEATS (the Knowledge Engineer's Assistant) is a software environment suitable for constructing knowledge-based systems. It differs from other shells and toolkits by providing support in the early stages of the knowledge engineering task. Its unique features include semi-automated transcript analysis facilities, and a graphical sketchpad from which code is automatically generated. These tools are valuable in speeding up the trial and error "massaging" of the knowledge which results in a structure. The design of KEATS has been closely tied to the ways of working we have evolved in practice (Motta, Eisenstadt et al, 1986).

In line with our policy of spreading the use of expert systems technology, the project will continue by adapting KEATS for more widespread use in Telecom, making it more accessible to the novice knowledge engineer. This will involve the addition of

a number of new tools:
- a graphical rule tracer
- a truth maintenance system
- a knowledge-base checker
- a learning component

Guidance on the best use of the system and on interviewing methodology will also be incorporated.

The system will also be provided with the ability to give specific, knowledge-based help to the knowledge engineer analysing a domain. It will incorporate a repertoire of common problem-solving paradigms employed by experts in a number of domains, which can be used to aid the structuring stage. This work reflects the value of domain models in the knowledge acquisition process. Work on identifying structures for diagnosis in a number of domains is already in progress.

## CONCLUSION

The study did succeed in extracting some information on the applicability and strengths of a number of knowledge acquisition techniques, but there is much to be added. There are very few ticks, and even fewer crosses, in the Table. Very few comparisons of methods had been made at the time of the study, although I am aware of some subsequent comparisons (eg Cooke and McDonald, 1986, Burton and Shadbolt, 1987).

More work on this subject would be very valuable. When taking on an expert systems project, especially in a business setting, the knowledge engineer feels that the expert's goodwill is essential and is anxious not to risk upsetting him. The tendency is therefore to be extremely conservative and to avoid any technique that might be thought childish and time-wasting. It is also important that any methods used should not require extensive training or practice.

This results in a reliance on techniques that are drawn from general business practice, such as interviewing and report-writing. For developers to exploit other techniques, guidance on their applicability and use must be available and incorporated into any tools that are employed, as is being done with KEATS.

## REFERENCES

ALVEY P L & GREAVES M F (1986)
Observations on the Development of a High Performance System for Leukaemia Diagnosis. Proc. Expert Systems '86, pp 99-110

BENNETT J S (1985)
ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System. Journal of Automated Reasoning, v1, n1, pp 49-75.

BOOSE J H (1984)
Personal Construct Theory and the Transfer of Expertise. Proc. AAAI '84, pp 27-33.

BREUKER J A & WIELINGA B J (1984)
Techniques for Knowledge Acquisition and Analysis.
Report 1.5 Esprit Project 12, Memorandum 2.8 of the research project "The Acquisition of Expertise".

BURTON A M & SHADBOLT N R (1987)
A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1
SERC Workshop on Knowledge Acquisition for Engineering Applications, June 1987, Abingdon.

CAVIEDES J E & BOURNE J R (1986)
Knowledge Engineering in Repair Domains: A Characterisation of the Task. Proc 2nd Conf on Expert Systems and their Applications, Avignon, pp 479-490.

CLANCEY W J (1984)
Knowledge Acquisition for Classification Expert Systems.
Proc ACM '84 Annual Conference pp 11-14.

COLLINS H M, GREEN R H & DRAPER R C (1985)
Where's the Expertise? Expert Systems as a Medium of Knowledge Transfer. Proc. Expert Systems '85, pp 323-334.

COOKE N M & McDONALD J E (1986)
A Formal Methodology for Acquiring and Representing Expert Knowledge. Proc. IEEE, v74, n10, pp 1422-1430.

De GREEF P & BREUKER J (1985)
A Case Study in Structured Knowledge Acquisition. Proc IJCAI '85, pp 390-392.

ELLMAN J (1987)
Practical Experience of Knowledge Acquisition for Expert Systems.
SERC Workshop on Knowledge Acquisition for Engineering Applications, June 1987, Abingdon.

FOX J, MYERS C D, GREAVES M F & PEGRAM S (1985)
Knowledge Acquisition for Expert Systems: Experience in Leukeamia Diagnosis. Methods of Information in Medicine, v24, n2, pp 65-72.

FREEMAN P (1985)
Knowledge Elicitation - A Commmercial Perspective. Proc. 1st International Expert Systems Conference, London 1-3 October, pp 339-349.

FRIELING M, ALEXANDER J, MESSICK S, REHFUSS S & SHULMAN S (1985)
Starting a Knowledge Engineering Project: A Step-by-step Approach. The AI Magazine, Fall 1985, pp 150-164.

GAINES B R (1986)
State ot the Art Report from the AAAI November '86 Workshop on Knowledge Acquisition. Presented at Expert Systems '86, Brighton.

GAMMACK J G (1987)
Different Techniques and Different Aspects of Declarative Knowledge. In A L Kidd (ed) Knowledge Acquisition for Expert

Systems: A Practical Handbook. Plenum.

GREEN R H & WOOD S (1984)
Expert Knowledge Elicitation - or How to Grow Trees. Proc. ES '84, pp 272-276.

HANNAN J J & POLITAKIS P (1985)
ESSA: An Approach to Acquiring Decision Rules for Diagnostic Expert Systems. Proc. 2nd Conference on AI Applications, Miami Beach, Florida, Dec 11-13 1985, pp 520-525.

INDER R, WAI HING CHUNG P & FRASER J (1987)
Towards a Methodology for Incremental System Development SERC Workshop on Knowledge Acquisition for Engineering Applications, June 1987, Abingdon

JOHNSON N E (1987)
Knowledge Elicitation in Practice.
Insight Study No. 6, Systems Designers Scientific.

MARCUS S, Mc DERMOTT J & WANG T (1985)
Knowledge Acquisition for Constructive Systems. Proc. IJCAI '85 pp 637-639.

MOTTA E, EISENSTADT M, WEST M, PITMAN K & EVERTSZ R (1986)
KEATS: The Knowledge Engineer's Assistant. The Open University Human Cognition Research Laboratory, Technical Report No. 20

REGAN J E (1986)
The Expert's Categories. Presented at Expert Systems '86, Brighton.

SHPILBERG D, GRAHAM L E & SCHATZ H (1986)
Expertax: An Expert System for Corporate Tax Planning. Proc. 2nd International ES Conference, London 30 Sept- 2 Oct pp 99-123.

SLOCOMBE S, MOORE K D M & ZELOUF M (1986)
Engineering Expert System Applications. Presented at Expert Systems '86, Brighton.

WELBANK M (1987)
Knowledge Acquisition Update. Insight Study No. 5, Systems Designers Scientific.

# On Structuring Knowledge for Incremental Knowledge Acquisition

Marc Linster
Institute for Applied Information Technology
German Research Institute for Mathematics and Data Processing
Schloss Birlinghoven
Postfach 1240
D-5205 Sankt Augustin 1
West Germany

## 1. Introduction

Domain independent knowledge acquisition systems have a drawback compared to domain specific systems, as they don't know what kind of knowledge to acquire. We propose to overcome this handicap by imposing inference strategy dependant structures upon the knowledge to be acquired.
We present a set of concepts that will help us structure and acquire knowledge for diagnoses purposes. They will be used in an architecture that both relies on and supports the structured acquisition of knowledge. A method specific to the acquisition of knowledge for diagnoses purposes is described in the last paragraph.

## 2. The conceptual framework

We define a conceptual framework for the acquisition of knowledge for diagnoses purposes. The elements of this framework, and their interaction represent the essential statement of our work. The definitions we give are not supposed to be valid outside the domain of knowledge acquisition, they shall only help us to state our point.
Our conceptual framework consists of **concepts, attributes, predicates, functions, symptoms, operations and rules.** The elements of this framework will help us structure the knowledge to be acquired.

- Concepts are the entities representing the domain.

  ex. : alarm-clock, spring, balance

- Attributes are properties with values, describing and refining concepts.

  ex. : motive_power(alarm-clock) = spring

- Predicates inquire about attributes and values of concepts

  ex. : motive_power_?(alarm-clock, spring)

- Functions are hierarchically organized properties of the domain, comparable to badly defined attributes.

  ex. : giving_the_time, ringing

- Symptoms are logical combinations of functions and predicates, possibly having a new name, thus providing complex constructs.

  ex. : bad_shape := to_late%time
  to_late%alarm
  cleaning_date_?(alarm-clock, long_ago)

- Operations change attribute values and are used in the right hand side of rules.

  ex. : wind_up(spring) *has effect* unwound(spring) := false

  wind_up(spring) is an operation, unwound(spring) an attribute, false is its value

- Rules are the connectives between the descriptive parts and the procedural parts of the knowledge base.

  ex. : IF bad_date_and_hour THEN set_watch

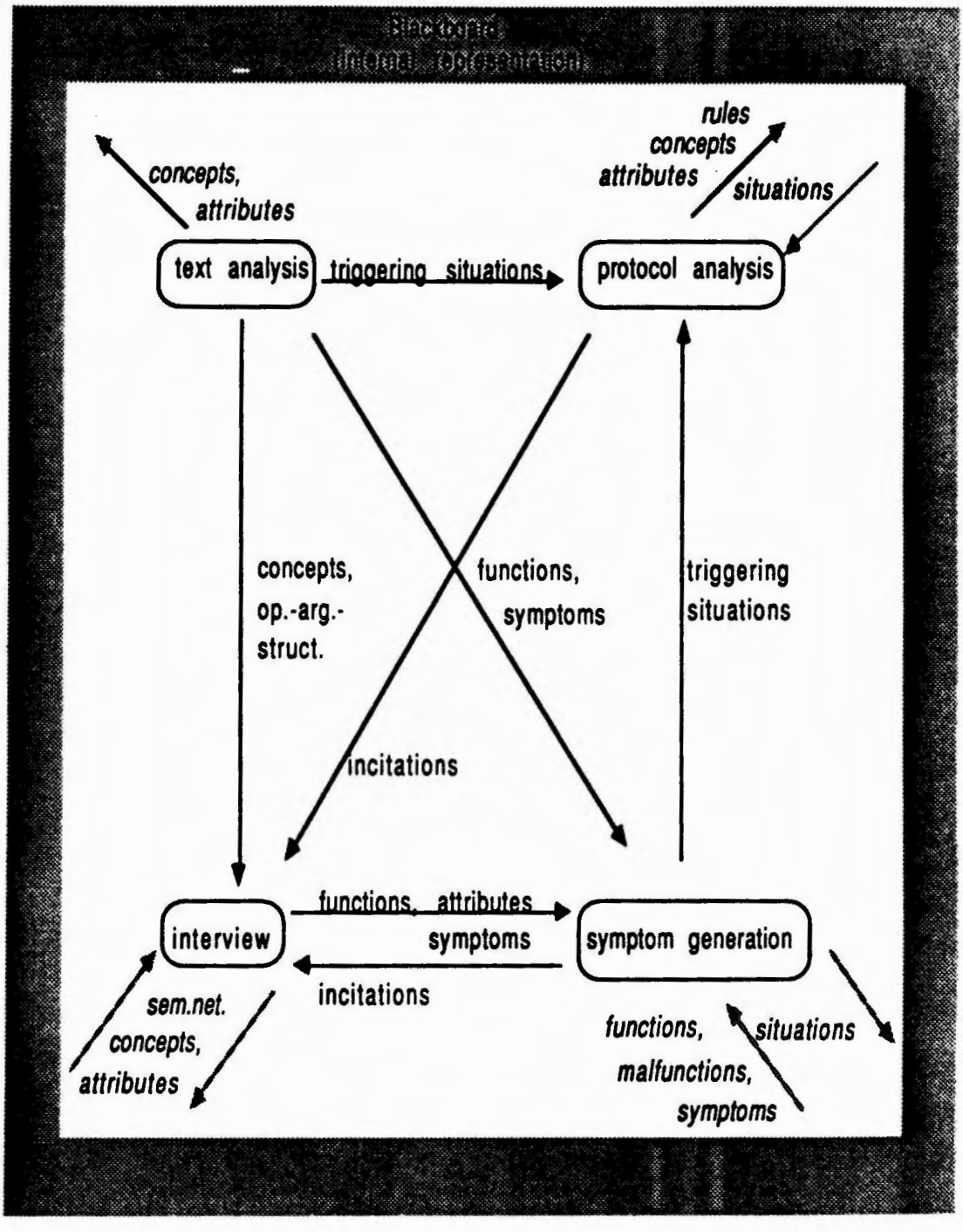
  IF unwound(spring) THEN wind_up(spring)

  unwound(spring) is a predicate, bad_date_and_hour is a function,
  wind_up(spring) is an operation.

## 3. Incremental knowledge acquisition in a hybrid architecture

A domain and inference independent knowledge acquisition system has to integrate a set of different knowledge acquisition modules, as all known acquisition methods show good performances in certain areas of knowledge, but have weak points in others. Interview techniques, like repertory-grid or laddering, are fit for the acquisition of knowledge for classification and differential diagnosis, procedural problem solving knowledge can best be elucidated with protocol analysis. A third acquisition method, text analysis is still of very limited use as a stand-alone-procedure, and can only be used to get an overview of the domain.
The task of integrating substantially different, but supplementary knowledge sources requires a special architecture. As all acquisition modules have a common aim, the acquisition of knowledge, a common datastructure seems best fit for the representation of the results. Every module uses the results of other acquisition modules, refines them and lets other modules use them. An area of "public domain" is needed, and together with the requirement for an integrating architecture it seems reasonable to introduce a blackboard architecture (Hayes-Roth 1978). A blackboard will help to integrate the acquisition methods, to manage their results and to control their access to the already acquired knowledge.
A hybrid knowledge acquisition tool should not just be an assembly of loosely connected methods, but it should be strongly based upon using the advantages of each technique in the domain that the technique is best at. The symbioses of several techniques seems best organized as a blackboard model.

Interaction of the acquisition methods



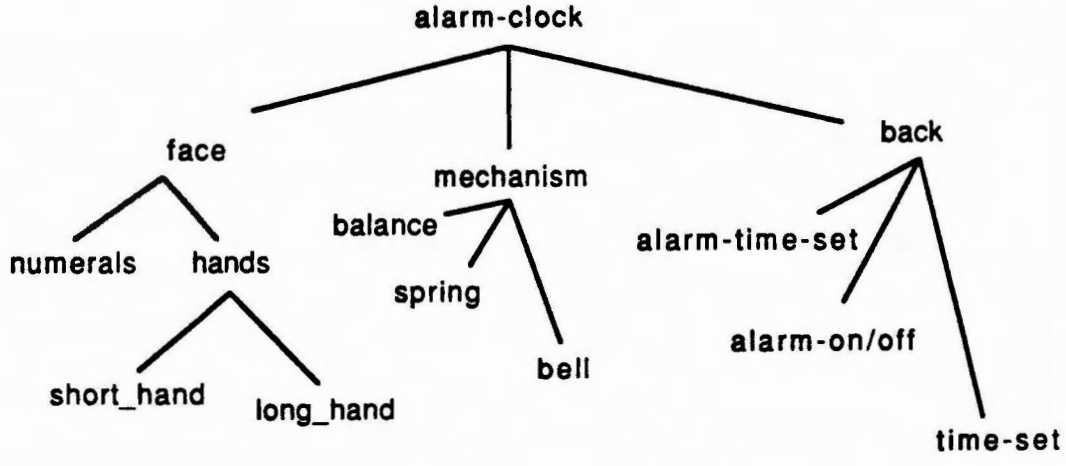

The different methods of a hybrid system depend on each others results (dark arrows), and they communicate through the blackboard (shaded arrows).

Text analysis may start an acquisition process by putting a set of unstructured concepts (objects, functions, malfunctions and symptoms) and constructs (attributes, operator-argument structures) on the blackboard. These again may be used by the interview component to build a taxonomic structure, add attributes to objects, refine them etc. The symptom generator reads functions and malfunctions from the blackboard, in the case knowledge for diagnosis is acquired, and writes symptoms onto the blackboard, which again can trigger protocol analysis. Protocol analysis feeds unknown concepts back to the blackboard, which may again trigger an interview.
This interaction allows for an incremental process of knowledge acquisition, each method pitching in what it is best at. As there are no conflicts between methods, because their ranges are distinct, we have no need for conflict resolution on this epistemological level.
Methods like symptom generation, but for other problem solving strategies, could easily be integrated into this organization, as blackboard architectures allow highly modular system design.

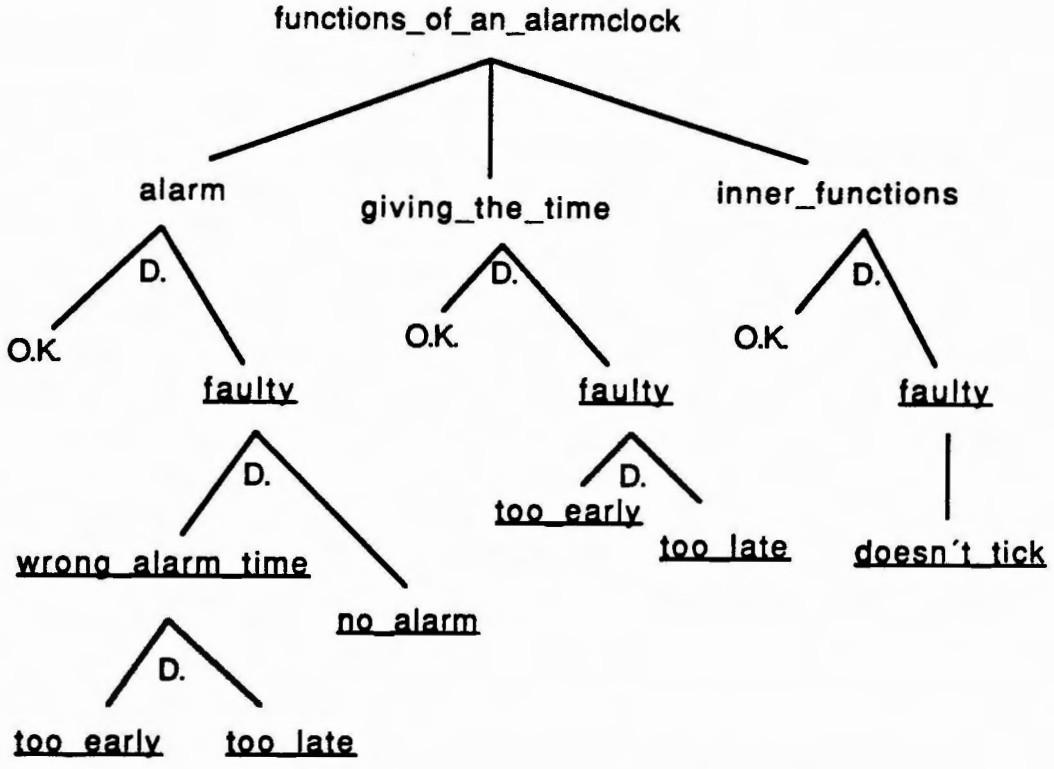
## 4. Example

A primitive alarm-clock will be used to explain the idea of structuring knowledge to support inference specific knowledge acquisition. Two diagrams will give a rough sketch of the concepts and functions that we will refer upon. The functions will be used to generate symptoms to trigger protocol analysis.

Parts of an alarm-clock:



Functions of an alarm-clock




Malfunctions are underlined, exclusive subtrees marked by D. (dichotomic).

## 5. Symptom generation

Protocol analysis is used to acquire procedural knowledge. Real situations in the experts environment or artificial laboratory situations are the starting-points for protocol analysis. In a diagnosis situation recognition and processing of situations are the tasks of the expert. Symptoms are the characteristics of diagnosis situations. If it were possible to generate all the symptoms a knowledge-based system might encounter and rate them by their relevance and probability, then protocol analysis could be used to systematically scan the needed procedural knowledge.
This would allow statements about the completeness of an expert system with regard to a domain, the domain being the static representation in the system, not the domain outside the machine.

### 5.1 Tasks and methods of symptom generation

We define symptoms as logical combinations of attributes, functions and malfunctions. As it is known if two functions can be active at the same time, and as we can give a rating of the probability that a malfunction happens, it seems possible to systematically generate symptoms. The technique we propose is domain independent, and because it is not yet based upon a deep model, every required information has to be acquired from the user.

### 5.2 Example of a symptom generation

The example will be based on the taxonomy "Functions of an alarm-clock".

Four kinds of questions are asked:
- *Can <function-1> and <function-2> be active at the same time ?*,
- *<function-1>, what will you do ?*, to trigger reactions to simple situations,
- *Do you expect new relevant information from the combination of <function-1> and <function-2> ?*, both functions describing subgraphs, whose members could be combined to form complex symptoms,
- *<function-1> and <function-2>, what will you do ?*, triggering reactions to more complex situations.

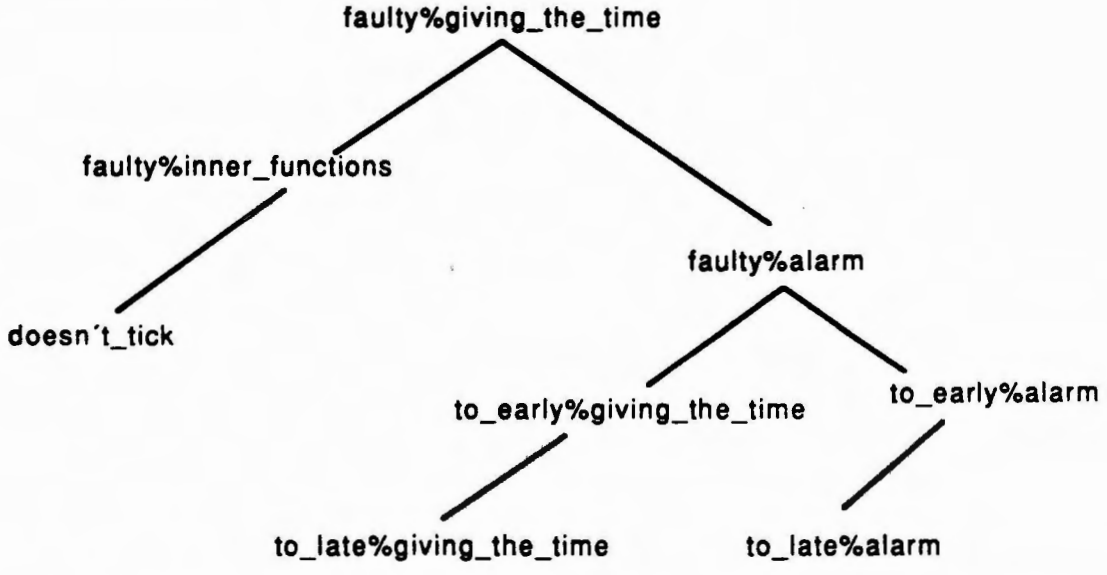
First the expert is questioned about the possibility that functions out of the same level of abstraction can be active at the same time.

*Symptomgenerator : Can "alarm" "giving_the_time" and "inner_functions" be active at the same time ?*
*Expert : yes*

*Symptomgenerator : Can "O.K.%alarm" and "faulty%alarm" be active at the same time ?*
*Expert : no*

Similar statements can be made about all the other functions of the taxonomy. The probability ratings of malfunctions do not have to be expressed numerically, a search tree would be a good enough representation to show which malfunction is more likely than another one.

Search tree of malfunctions




In the graph representing the possible malfunctions, leaves are used as basic situations to trigger protocol analysis. These protocols can be retrospective verbalizations, protocols from memory, or the malfunctions are used to build real situations that the expert has to work in, and produce thinking-aloud protocols (Ericsson, Simon 1980).

*Symptomgenerator : To_early%alarm, what will you do ?*
*Expert : To_early%alarm and O.K.%giving_the_time then check the alarm time setting.*

*Symptomgenerator : Doesn't_tick%inner_functions, what will you do?*
*Expert : Doesn't_tick%inner_functions and not unwound spring and price of alarm-clock low then trash it.*

In a second phase subgraphs described by functions that can be active at the same time will be combined to form more complex symptoms.

*Symptomgenerator : Do you expect new relevant information from the combination of faulty%alarm and faulty%inner_functions ?*
*Expert : no*

*Symptomgenerator : Do you expect new relevant information from the combination of faulty%alarm and faulty%giving_the_time ?*
*Expert : yes*

...

*Symptomgenerator : To_late%alarm and to_late%giving_the_time, what will you do ?*
*Expert :*
*1. To_late%alarm and to_late%giving_the_time and wound up spring then clean mechanism.*

*2. To_late%alarm and to_late%giving_the_time and not wound up spring then wind up spring.*

The probability ratings of the malfunctions are supposed to control the process of symptom generation if there are larger sets of malfunctions.

### 5.3 Discussion

The most important aspect of the symptom generation is its capability to systematically scan the field of probable situations that the expert system may be up against. As long as it is not based upon a deep model the symptom generation cannot be complete, nor can it be efficient. But at least it forces the expert to deal with a wide range of possible cases.

## 6. Conclusion

Such a framework, not necessarily identical with the one we proposed should support knowledge acquisition for specific problem solving strategies in an efficient way. Future work will have to show how useful, and how adequate such structuring is, and if similar frameworks can be defined for other problem solving strategies.

## 7. References

Diederich J.
Verfahren des Automatischen Wissenserwerbs
GMD Jahresbericht, St.Augustin1986

Ericsson K.A., Simon H.A.
Verbal Reports as Data
Psychological Review, Volume 87, Number 3

Hayes-Roth, P., "Understanding Speech in the Hearsay-II System" In : Speech Communication with Computers, Editor L. Bolc, Hanser, Wien 1978, P 9-42

## 8. Acknowledgments

Thanks to Joachim Diederich, Thomas Uthmann and Joachim Hertzberg who read and commented drafts of this paper.

# Teaching Knowledge Engineers the Human Side of Knowledge Engineering

August 1987

Jill Bell

JB Associates
47 Western Road,
Winchester,
Hampshire,
SO22 5AH

## THE PEOPLE AND THE COURSE

I will briefly introduce the instructors, the students and the course content.

### The Instructors

Bob Hardiman has been involved with computers for 20 years, works in the education department of his company and was doing post-graduate research at Bath University when he had the idea for the course. Bob and I had worked together teaching personal development for some years, so he came to discuss the idea with me. I became involved with some trepidation, because, from where I was standing, it seemed a very technical area that I was being drawn into.

My background is completely non-technical: I had only ever involved myself with people and at that time, just over a year ago, I was computer-phobic. It soon became clear that here lay my value in developing the course "The Human Side of Knowledge Engineering": the field of knowledge engineering is a meeting place for people and computers. Bob was sufficiently steeped in computer technology that

he tended to take for granted a whole range of things which were a mystery to me. My naive questions and paralysis in the face of anything too technical served as a constant reminder of what "user-friendly" really meant, and of how vitally important the human issues were.

## The Students

The students have come on this course from very varied backgrounds, from fifteen different companies so far, and from all over the world. We have taught it nine times to date in the USA, Germany, UK and Singapore, and are booked in the UK and California for four more courses this year, and another series in 1988.

Some students from technical backgrounds have found the human skills content difficult. All the students have been a rich source of ideas and constructive feedback from which the course has gradually evolved.

There has also been a wide variation in students' experience of expert systems and in their expectation of future involvement in that field: they have ranged from those coming along to get themselves informed for the benefit of staff or customers, to those due to start a project the following week.

## The course

The course lasted three days when it started out -- with notes consisting of smartly bound blank pages with the name of the day at the top of each. It felt like the blind leading the blind where they didn't want to go. The course now lasts four days and the notes have expanded to 70 odd pages plus illustrations. The first two days consist of teaching techniques, practical exercises and discussion; the third day is devoted to a case study; and on the fourth day teams present their case study findings and we fill in the background theory.

The course developed from a recognition that a great deal of research had been done at the computer-end of knowledge engineering, but very little at the human end. The course is experiential and aims to answer the question "How do you get the knowledge out of the expert?" Often the people involved in doing knowledge engineering have previously been involved with machines, and black and white facts. They have been used to linear, "tidy" thought processes and are sometimes at a loss when confronted by human beings with their inconsistency, their "messy" knowledge, and their even messier feelings. The course sets out to chart this territory, to provide practical guidelines, special techniques and a clear structure for the knowledge engineering process. The methodology is described more fully in (Hardiman, 1987; Bell, 1987), but I will list the key points and comment briefly on each as a background to what I have to say.

## KEY POINTS IN THE PROCESS

- KE is a Cooperative Process
- KE is Research with People
- KE works in the Natural Setting
- KE uncovers Tacit Knowledge
- KE is an Iterative Process
- KE needs Auditing
- KE produces Context-Sensitive Results

## KE is a Cooperative Process

Knowledge Engineering is a team activity. The team should include user(s), expert(s), managers and knowledge engineer(s).

### The User

The user comes first because he has so often been neglected at great cost. Frequently the reason for the failure of an expert system is that the user was not consulted at the outset, or at all. By a failed system I mean one which may work fine but which simply isn't being used. Sometimes the system is solving a problem which the user didn't have in the first place. Sometimes it transfers the problem from the expert to the user, and in the process transfers responsibility which the user does not want, and for which he is not being rewarded.

It is also important to find out what degree of sophistication the user has in the system's domain, in order to aim the system at the right level.

### The Expert

Ideally knowledge engineering requires an enthusiastic, articulate expert with plenty of time to spare. Unfortunately these don't exist: if they're that good, they won't have any time to spare. Indeed, it is probable that the reason for developing the system is that the expert is currently so stretched. However, it is very unlikely that you will manage to develop a useful, working expert system unless you have wholehearted commitment from the expert, and that includes a commitment of time.

The expert should also be a real expert, not someone who learned his skill in two weeks and has been repeating it for the last 25 years. The expert will be someone who has developed unique ways of working, has his own, idiosyncratic, knowledge and who is a scarce resource within the company.

### Management

The user's, the expert's and the knowledge engineer's managers need to be fully involved. One of them must take responsibility for managing the project. This involves allocating man-hours and budget, and deciding who shall have responsibility for maintaining and updating the system when it is in use. It is also necessary to realise that matters such as settling issues of new responsibility which the system may bring will have to be dealt with. The message for managers is that an expert system project needs as much managing as any other, and will not be a swift affair.

### The Knowledge Engineer

The personality of the knowledge engineer is important: very shy or very aggressive people do not make good interviewers, especially if the expert or user is shy, confused or resistant. Nor do people with fixed ideas or purely technical backgrounds. They may also need a rare mixture of persistence and diplomacy for the task of stage-managing co-operation between managers, users and experts.

First and foremost the knowledge engineer needs to be a good listener. She also needs an open mind and a sense of humour. Counselling skills will be more useful than programming skills. Some of the more well-known knowledge engineers were more psychologists than technologists, although people with a blend of human and technical skills will fit the bill best of all.

## Knowledge Engineering is Research with People

### Research

The KE process sets out to explore areas of human knowledge -- at least one (the expert's), preferably two (the user's and the expert's), probably more (multiple expert or multiple users). This is precisely what social scientists do, and so we have borrowed from their research methods to develop our method of knowledge engineering. This is described by Bob Hardiman (Hardiman 1987)

## People

People have feelings and prejudices, good days and bad days. They are not predictable or even consistent. It is common for an expert, however articulate, not to know how he does what he does. All this makes difficulties for the knowledge engineer -- who also has feelings and prejudices, good days and bad days.

## People have Feelings

Some people, especially in the technical field, tend to dismiss feelings as a nuisance and/or irrelevant. In our experience they are very relevant to knowledge engineering and only become a nuisance if they are ignored. If they are acknowledged they can become a powerful tool. If they are ignored essential information may never come to light: information which may be critical to the success or failure of the expert system.

Take, for instance, an expert who does not properly understand the purpose of the system, and who may have fears of losing status or becoming dispensable. If these fears are recognised and respected, and time is taken to explain that the system will take care of tiresome and repetitive routine problems, leaving the expert free for more important work, he may be much more inclined to co-operate with the knowledge engineer.

This may seem a very obvious thing to do, but it is surprisingly easy to neglect.

It is generally accepted in the field of humanistic psychology (Maslow 1968) that human beings have three fundamental emotional needs (apart from physical, intellectual and spiritual needs), which are matched in the workplace, by three parallel needs:

- love -- appreciation
- understanding -- good communication, clear objectives
- freedom -- self-direction

If we follow Maslow (and look into our own life experience) we find that frustration of our needs for love, understanding and freedom produce negative emotions. Similarly if people at work (and in a knowledge engineering project) are continually denied appreciation, good communication and self-direction, they will experience matching negative emotions:

- grief -- apathy
- fear -- anxiety
- anger -- hostility

These negative emotions are likely to sabotage the project, often in subtle ways -- people's interest may flag, they will become elusive and unavailable, and information will be witheld.

Which is a good argument for a knowledge engineer to:

- give recognition to everyone involved (appreciation)
- keep them informed (good communication)
- involve them in the planning (self-direction)

If she does this the other side of the coin will appear, which is the normal state of affairs when these fundamental needs are fulfilled: participants will become:

- highly motivated
- confident
- creative

Well, in an ideal world they will!

This summary does not allow for the complexity of human feelings and behaviour, but I think it is a useful guide.

## Knowledge Engineering Works in the Natural Setting

What people actually do usually differs from what they say they do. Not because they set out to deceive, but because the sorts of skills that we are exploring in the KE process are not easily articulated.

For this reason it is most productive to observe the user or expert in the place where they usually go about their work. Watch them actually doing it. This will yield different information from interviews away from the workplace. These have their value especially if the workplace is crowded or noisy, but should not be solely relied upon.

Of course the presence of the knowledge engineer changes the natural setting and this cannot be avoided. But be as unobtrusive as possible and be aware that your presence may affect the expert's or user's behaviour.

## Knowledge Engineering Uncovers Tacit Knowledge

Tacit knowledge poses a particular problem for knowledge engineering, however willing, committed and articulate the expert. It is knowledge that may be difficult to elicit for one or both of two reasons:

- It may be knowledge which was once learnt but so long ago or at

such a fundamental stage that it is too deeply embedded to access easily -- procedures which are carried out almost unconsciously, on "automatic pilot".

- It may be expertise which cannot be articulated; only acquired by experience, trial and error. Take the skill of riding a bicycle: all the necessary mechanical movements can be described without conveying the essential "how".

Close observation and skillful questioning may uncover this kind of knowledge (see the section on the natural setting, above), but we also teach some specific techniques which are designed to access it -- incomplete systems, 20 questions, repertory grid, critical incident analysis, and others.

We also need to include under the heading of tacit knowledge the user's latent awareness of his needs which only crystallize when possible solutions are presented to him.

## Knowledge Engineering is an Iterative Process

The research process involves going out and collecting data, bringing it back and looking at it, sorting and analysing it, looking for patterns, seeing what's missing or what comes next, and going back to get more. This cycle continues until a complete picture emerges, or until the time or money run out.

This process is vital to the knowledge engineer's evolving understanding of the data, and even more vital to the formulation by the user of his needs which may well only emerge and develop in response to the emerging possibilities.

## Knowledge Engineering Needs Audit

Because people, including knowledge engineers, have personal biases, motivations and blind spots, the outside auditor of a knowledge engineering project carries an important role.

He should not have any investment in the project, no personal axe to grind, although he should have a grasp of the knowledge engineering process. He should not be any of the managers involved, and preferably not a manager at all.

His job is to bring a fresh perspective to the work, and to ask awkward questions about what has or has not been done and why. As with a financial auditor, his concern is twofold: looking at the data to see if it has been processed correctly, and looking at the process of the inquiry, to see that it has been carried out in a trustworthy fashion.

## Knowledge Engineering Produces Context-Sensitive Results

An expert system is not necessarily transferable from one location to another. People and their circumstances vary from place to place even in the same company, and their needs and ways of working will vary too. In most cases, expert systems will only be of use in the department where they are originally conceived. Even other departments with very similar functions will probably need to modify the system for their own use.

For this to be possible, the KE process should be well documented to allow others to judge the applicability of the system to their own contexts. And they will still need to go throught the process of knowledge acquisition before they are in a position to see if there is a reasonable match.

## KNOWLEDGE ACQUISITION TECHNIQUES

The techniques fall into three broad categories of knowledge collection, analysis and representation, which necessarily overlap during the cycles of the iterative process.

### Knowledge Collection

We refer to any face-to-face encounter of the knowledge engineer with users or experts as an interview. Interviews fall broadly into two categories -- open interviews which use general interviewing skills (mostly about listening) to establish rapport, understanding and the big picture; and focused interviews which use specific techniques to gather facts and detail. These are described in the course notes (Bell 1987).

### Knowledge Analysis

This is the process of breaking up the data collected into individual stand-alone units (chunks) which are then sorted into categories (piles). The naming of categories should be postponed for as long as possible so that a pattern or structure emerges from the data rather than being imposed upon it by pre-conceived ideas. Relationships and hierarchies are sketched in, and the holes in the pattern and the missing links are used as a basis for planning the next stage of iteration.

The process is partly intuitive and benefits from some suspension of left brain activity -- if only to gain distance and see the wood for the trees, as in "let's sleep on it". Unfortunately a four day course doesn't allow for this option.

## Knowledge Representation

We deal briefly with five methods on the course, and make two points. The first is that a knowledge representation method is effective if it makes sense to all concerned and can be simply explained to an outsider, eg the auditor: it doesn't have to be sophisticated or by the book.

The second point is that it is required to be understandable and computable -- understandable first and foremost, otherwise it is no good as a tool for further knowledge acquisition, and will prevent involvement of the team in the development process.

## SPECIAL PROBLEMS

I would divide these roughly into task problems and process problems. They are problems which the students have had, problems I have had as an instructor, and problems shared between students and instructors.

### Task

Task problems have included lack of interviewing skills; panic about knowing what to do with the quantity and chaotic nature of the data generated in interviews; a tendency to home in too quickly on reassuringly factual detail without understanding the essential context first; difficulty with understanding the process of data analysis (which was largely to do with the process problem of how we presented it); not knowing which techniques suited which type of knowledge domain; and, above all, not having enough time. This is common enough in real life projects, but is exaggerated on a one-day case study.

### Process

Process problems have been about the expectations of the course, about the difficulty of letting go of traditional methods and trying a new approach, and about trying to put new techniques into practice back at the ranch and sell it to sceptical or hard-pressed managers.

As instructors we have tried to find a balance which gives students enough space to make discoveries for themselves without leaving them floundering too long in the unknown. We have not always got this right: we have presented the class with too much theory too early and had it thrown out as so much jargon; students have been at a loss for lack of structure; and there have been cries of "but how do you actually DO it?"

We have responded by reducing the theory and putting it at the end of the course; by introducing more structure as we saw the need, and as our confidence grew; by giving practical demonstrations of the problematic techniques; and by introducing more practical exercises throughout the course with feedback and discussion sessions after each.

## Culture Shock

Two things have alerted us to the cultural difference between our course and other courses. One is the amount of furniture removing we have to do after the room has been set up "normally", ignoring our request for layout. The other is the way students look in, walk past and then come back asking tentatively "this isn't the knowledge engineering course is it?"

We seat people in a circle of chairs with the tables round the edge of the room so that students can turn round and use them as necessary. This involves the least amount of hardware coming between the people in the room, but may leave students feeling exposed without their customary defences (table, briefcase etc.). So we spend a lot of time over the introductory session so that the group begins to feel safe: members start wanting to get to know each other and work together rather than feeling threatened by each other.

## Demonstrations

Demonstrations are worth a special mention as they turned out to be an instant success in two areas where other forms of explanation had made no impression.

The first area was repertory grid which students have found one of the most useful (and compulsive) of the data collection techniques. Our careful foil presentation left them puzzled, but when they were able to gather round a PC and watch me build a grid the scales fell from their eyes. We use a repertory grid program developed by Bob Hardiman based on the ideas of Shaw (Shaw, 1980). Students have also given us useful feedback on the separate advantages of triadic and dyadic elicitation (see Keene and Bell in Shaw, 1981).

The second area was in our method of qualitative data analysis which is so simple that it eluded people until they saw it demonstrated. We had a second breakthrough here after we arranged the exercises so that students transcribed and analysed their own data. Previously they had practisedon a ready-made transcript of a video-taped interview which was not nearly so successful.

On the last course we taught it was suggested that we should demonstrate a whole sequence of the process -- from open interview, through the chunking and sorting of analysing the data collected, and possibly including the knowledge representation which emerged. This

is close to an idea we had early on for making a video of this process. We plan to have another attempt at improving our video skills and follow up that idea.

## Tolerating Chaos

A certain amount of chaos is inevitable in the KA process because of the volume of the information collected and the unordered way it typically falls from the lips of users and experts. (After being in the role of expert on four separate occasions on the same subject I was still presenting my expertise in a jumbled mess, although on the last occasion I fondly imagined that it was organised and tidy.)

There is a tendency on the part of interviewers, who are panicking about volume of data and time running out, to start trying to shut the expert up or direct the interview inappropriately. The role of the observers and feedback and discussion groups has been invaluable in demonstrating that these interviewing tactics really did antagonise the expert especially in the early stages.

## Transitions

The chaos described above tends to produce a lot of panic, depression and anger in students, particularly in those who have been used to working systematically with black and white facts. This distress has been most apparent in the area of knowledge analysis and in trying to make the transition to a knowledge representation with rules and a structure on which to build the system. The cry goes up for a clearly defined, step-by-step methodology. My feeling has been that there isn't one, but I am now willing to be proved wrong; or to think that if there isn't, then I need to invent one or find one.

This issue has been complicated by our confusion about how much of the chaos is intrinsic to the process and unavoidable, and how much we have put there or failed to eliminate by not providing a clear methodology for that transition. As things are, students are being asked to make a leap into the unknown relying on intuition -- small wonder that they want some stepping stones, especially if they have a hearty mistrust of intuition.

Most people seem able to accept that a "soft"/intuitive/passive approach is suitable for the social process of interviewing, but have great resistance to suspending the "hard"/objective/dominant activity of the left brain when it comes to the mental process of knowledge analysis. On the other hand most people have noticed the value of sleeping on a problem at some time in their lives. However, it is important that we work with techniques we feel comfortable with and for those who cannot cope with active right-brain techniques like drawing, fantasizing, meditating or creative thinking (van Oech 1983), I suggest that they at least "switch off" by going for a walk or to the pub or by putting a night's sleep between themselves and the

problem.

Throwing Away

This problem is closely connected to the last two: in the process of wading through a jungle of information there will inevitably be false starts, blind alleys and work done which is of no use in itself to the end product. Groups have understandably found it difficult to throw that work away (we advise against actually throwing it away because they may change their mind and want it back) without feeling that it represents a waste of precious time. Throwing away is an essential part of the iterative process of learning about the field: there may be nothing concrete to show for it but the exercise will have advanced the understanding.

## MESSAGES FROM THE CASE STUDIES/COURSES

One of the many pluses of teaching this course has been the lively participation of the students and the constructive feedback we have had from them. From the outset we made it clear that we needed to learn as much from them as they from us -- and I think we have. One thing that became clear during the early courses was that most of the learning happened during the case study when students were applying the new techniques. Noticing this, and responding to demand, we put in more exercises earlier in the course. As a result new techniques become meaningful sooner, and there are more learning opportunities for everyone (including ourselves) as the groups report back after each exercise.

Teamwork

Over and again we have heard students say that they couldn't have undertaken the work without the team. Some reasons for this are task related, some process related.

Concerning the task, the team made it possible for users, experts and managers to be in communication with each other and with the KE. The input of each was essential for the project to be usable, reliable and viable. Also in a team more skills were available, and more labour -- larger teams tended to get more done.

Concerning process, the team met the need for moral support -- to cope with the volume and complexity of the data. It was also found invaluable to have back-up people observing, recording and contributing in interviews when the interviewer's attention was saturated. Different points of view within the team were also useful, and members were able to pool knowledge and benefit from a synergy of ideas.

If groups wanted to be critical of the team approach I noticed they referred to it as a committee -- and this happened exclusively and frequently with reference to knowledge analysis. Given the intuitive content of the process this is not surprising and coincides with the view of Judi Marshall of Bath University who cannot imagine sharing the task. Groups have been inventive about solving this problem, some splitting up to do it and having various strategies for deciding which interpretation to go with in the event of a dispute.

## Open and Focused Interviews

The teams strongly endorsed our insistence on doing open interviews first, allowing the expert to talk freely and sketch in the "big picture", to establish a context for the detailed information which emerges later. Some students were sceptical about this at first, but subsequently found they couldn't manage without this approach.

## Asking the User

We make a particular point of the need to consult the user, and stress that expert systems fail if the user is not involved in their development. But in spite of this it is rare for groups to think of interviewing the user in the early exercises: over and over again in the feedback sessions we hear the message that users feel overlooked and unimportant. Later in the course, when we think the message has been heard, users continue to report that their needs are not being met and that only lip service is being paid to them. There are three points to be made here.

The first is that we all need to tear ourselves away from the experts and the problem of how to get the knowledge out of them and apply the same or similar techniques to getting the needs out of the user. Otherwise it is a hollow gesture to involve him or her.

The second point is that the user's needs are potentially as tacit as the expert's knowledge, but in a different way and for different reasons. Take the typist who has never seen a word-processor. How can she imagine what benefits such a machine might bring when she has only ever known a typewriter? The user's needs will often be unknown until he or she knows what is on offer; then they will evolve as the possibilities emerge during the iterative process. As soon as the typist understands a word-processor she will start applying it to her own work and thinking of changes and refinements to suit her own needs.

The third point is that users need to be assertive. Often in a company they are lower in the hierarchy than the experts, management or the knowledge engineer, so it may require considerable courage to speak out. On our course they are part of a peer group, and yet they still experience difficulty in asserting themselves sufficiently to be heard and maintain a point of view.

It seems that the status of users on the course reflects their status in real life. We need to work hard to change that by applying the points made above. We need to enhance the status of users by putting them in the limelight; teach knowledge engineers how to encourage and support users; convince managers that users are important enough to be released from doing the job to be involved in the project; and teach users the skills to handle the situation -- otherwise the limelight will feel like a very uncomfortable hot seat.

## Analysing the Knowledge

We are still finding out how to teach this. With some groups it has gone well and they have had no problem; with others we have failed to press the magic button.

We have learnt that you can't analyse someone else's data satisfactorily.

We have learnt that if you are transcribing a tape-recording it is worthwhile to do it yourself.

We have also learnt that transcription is a very time-consuming process and that it is unrealistic to expect everyone to do it. It may be replaced by taking notes from the recording; but there need to be safeguards against leaving things out.

Equipment helps -- lots of little yellow sticky notepads, scissors or, if possible, a metaplan system which consists of floor-standing boards and cards of all different colours, sizes and shapes to pin onto them.

It has also emerged that analysing data isn't a suitable group activity. Each person tends to chunk and sort in a very individual way and this leads to disputes and wasted time.

## Representing the Knowledge

Many teams were apologetic and dismissive about the methods they used to represent their findings because they seemed messy or childlike or did not conform to some imagined "proper" standard. We came to the conclusion that any KR technique is valid and successful if it is understandable to everyone involved and demonstrable to an outsider: in the case-study, to the rest of the class, in real life, to an auditor and to management. It is also necessary that a mapping can be found between the understandable KR paradigm and the target computable method.

## The Transition

The transition is from the analysis to the representation of the knowledge. The question that tends to be asked is "how do we get from here (chunks and piles) to there (structure, rules)?" This was often the point at which we, the instructors, would be called in to a case study group to advise on what to do next. We have not always been very helpful, sometimes because we wanted students to discover for themselves, but sometimes because we didn't know. Answers are evolving from what students tell us of their experience and from our own practice.

Some answers are about letting it happen: "we went to lunch and it seemed much clearer afterwards"; or "we were talking about something completely different and it suddenly came to me". These answers are letting the right brain have a go at tackling the problem, as we do when we stop trying consciously to remember someone's name and it comes to us "out of the blue" as we relax in the bath.

Other answers are about developing a methodology. And the most vital which has evolved is, yet again, to turn to the user to find the solution to this problem, for this reason:

Expert systems are simple and trivial. Expert knowledge is complex and involved. So that after data-gathering sessions with the expert some teams felt overwhelmed by the sea of information which they had to fit into one small pint pot.

The users's view, on the other hand, tends to be simpler, more direct and more pragmatic. Time after time teams have found that the user's needs have provided the structure and the design of the system -- the skeleton on which to hang the flesh of the expert's knowledge from which the actual rules will be derived.

## Encouragement

Because knowledge engineering is a new area and involves new ways of working for many students, they need reassurance that they are on the right lines and encouragement to try out new techniques. This is true on the case-study when most groups welcome visits from the instructors to see how they are getting on. (They have been known to drag us in and lock the door.) And I have also found it true during consultancy: in one company it was necessary to hold the hands and jog the memories of the KE team as they struggled to transplant what they had learnt on the course into the less favourable environment back in their place of work.

## IN CONCLUSION

### Ideas for the Future

As well as the course referred to here I now offer separate packages for knowledge engineers who want to develop skills in particular areas -- assertive communication, interviewing skills and working in a team.

My co-teacher, Bob Hardiman, is working on the idea of a course which leads on from the human to the technical side of the process and takes the knowledge engineer through the stages of developing a system for real, using a disciplined, formal process.

We both feel a need to develop a methodology for knowledge analysis and representation and are working on it. Writing this paper has helped ideas to flow, and I now have the urge to write a paper entitled: "Method and Madness in Knowledge Analysis".

### The Process Works

When I first became involved in KE I was surprised and alarmed to find myself there, in what felt like a very technical environment. I wasn't sure that I had a contribution to make and found it difficult not to collude with the students who didn't want to look at the human side of the process. Since then I have not only become confident in my role, but have acquired an understanding of some of the technical side and have built prototype systems myself. As I gain more experience of the whole KE cycle I become even more convinced of the critical importance of giving human issues priority.

I find too that the iterative cycle of action and reflection works; that I need to change focus from the big picture to fine detail and back again to keep a balanced perspective; that building prototypes and our many other knowledge elicitation techniques work.

Teaching knowledge engineering is in itself an iterative process: as we learn from the students we adjust the course and the notes accordingly. I also frequently feel when students break through into a new dimension of understanding on the last day of the course that it would be useful to go through the whole process again with that fresh perception.

There is a saying that the best way to learn about a subject is to teach it. This has certainly been true for me in the field of knowledge acquisition and I would like to place on record the debt I owe to all the students on the courses. I look forward to continuing to develop that knowledge with future students.

REFERENCES

Bell, J., (1987) Notes to Accompany the Course "The Human Side of Knowledge Engineering" JB Associates, Winchester, UK

Boose, J H (1985) A Knowledge Acquisition Program for Expert Systems based on Personal Construct Psychology International Journal of Man-Machine Studies Vol 23, Number 5

Hardiman, R. J. (1987) A Naturalistic Approach to Knowledge Engineering School of Management, University of Bath, UK

van Oech, Roger (1983) A Whack on the Side of the Head Creative Think, California

Maslow, A.H. (1968) Toward a Psychology of Being Van Nostrand Reinhold, New York

Shaw, M L G (1980) On Becoming a Personal Scientist Academic Press, London

Shaw, M L G (1981) Recent Advances in Personal Construct Technology Academic Press, London

# A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1 (*)

A M Burton, N R Shadbolt, A P Hedgecock and G Rugg
Department of Psychology
University of Nottingham
Nottingham NG7 2RD

**Abstract**

Four knowledge elicitation techniques are compared in an experimental study. The techniques are interview, protocol analysis, laddered grid and card sorting. Knowledge bases were elicited from 32 experts, each being subject to two techniques. Protocol analysis is seen to take longer, and yield less information, than comparable techniques. Laddered grid and card sorting, despite being contrived and unusual techniques, are shown to be as efficient as more traditional techniques. Data are presented relating experts' individual characteristics (personality and cognitive style) to the efficacy of each technique. The possibility of matching experts to knowledge elicitation techniques is discussed.

Keywords: knowledge elicitation; knowledge acquisition

(*) This work was carried out as part of an ALVEY award to the authors (IKBS 134). We would like to thank Prof. Baker, Dept of Geology, University of Nottingham, and his colleagues for their kind help in preparing the stimulus materials for this study.

## Introduction

A major stumbling block in the development of Expert Systems is the process of getting knowledge from a human expert (Hayes-Roth et al, 1983). The problem of knowledge elicitation (KE) is an applied problem in Cognitive Science embodying as it does issues of human memory and artificial systems of knowledge representation. In this paper we present an experimental study of the efficacy of various established techniques.

The literature on KE is disappointing for those wishing to build an Expert System. Most studies of the effectiveness of various KE techniques fall into two categories: (i) those which report a case study; and (ii) those which attempt to match KE techniques with types of knowledge on theoretical grounds. The first type of study (e.g. Smith and Baker, 1983) is unsatisfactory because one may not generalise the results. The efficacy of the various techniques will very probably vary across domains and even across individual experts. Moreover, the recommendations that emerge, such as "don't forget to record your interview", leave one with the feeling that there ought to be more to say. The second type of study (e.g. Gammack and Young, 1985) tends to be uncompelling. Certainly the mapping of KE techniques onto domains often seems plausible, but then alternative mappings seem equally plausible. For instance, it is all very well to recommend a procedural technique when tapping procedural knowledge, but how is a novice meant to decide what is procedural and what declarative for a given knowledge domain?

This distinction, between procedural and declarative knowledge has been used a great deal in KE work. Varieties of the distinction can be traced through the work of a number of philosophers and psychologists (Ryle 1949, Anderson 1983). The essence of the distinction can be captured in the difference between *knowing how* (procedural knowledge) and *knowing that* (declarative knowledge). Different techniques in KE are seen as differentially tapping procedural and declarative knowledge.

Our dissatisfaction with the KE literature led us to undertake a study evaluating KE techniques.

We have reduced this study to a laboratory experiment. Of course this may incur scaling problems, but this risk seems worthwhile considering the gain we make by being able to generalise results from a multi-expert study.

**The techniques**

The four techniques we have chosen to evaluate in this study are:

1. Formal Interview: This is the standard common sense KE technique. The interview is formal in that its structure is closely specified. The format used in this study involves the expert providing an overview of the task, followed by the elicitor seeking verification of particular assertions in a recursive manner. The elicitor is only allowed to use certain probes, and the whole process is directed at eliciting rules, rather than vague recommendations etc. This is allied to Grover's (1983) 'forward scenario simulation'. During the interview hypothetical problems arise, and are solved by the expert. The formal interview primarily elicits procedural knowledge.

2. Protocol Analysis: In this technique one observes the expert actually solving problems in the domain of interest. The expert is encouraged to verbalise decisions taken on the way to a solution. Rules are induced in a post hoc manner from the surface behaviour of the expert. Protocol Analysis primarily elicits procedural knowledge.

3. Goal Decomposition (Laddered Grid): In this technique the problem space is represented as a hierarchy of goals - terminating in solutions to the problem. The elicitor makes a random entry into the space and moves around it with prepared probes to explore up, down and across the hierarchy. The space is built up with pencil and paper in front of the subject. As an example, imagine the domain is the classification of animals - one may enter the domain with the target "dog", and use such probes as "what examples are there of dogs?" in order to move down the hierarchy. There are other versions of goal decomposition (Grover 1983), but the laddered grid provides a suitable example of the generic technique here, because it is easy to

hold constant for many KE sessions. A laddered grid primarily elicits declarative knowledge.

4. Multi-dimensional analysis (Card Sort): This technique provides a qualitative multi-dimensional mapping of the elements in the problem domain. This is achieved by repeated sorts of a deck of cards, marked with the domain elements. On each sort, the expert is required to label the overall scale (factor) and the individual piles (levels). Rules are then extracted through classification matches. The card sort primarily elicits declarative knowledge. (*)

These techniques are not designed to be used alone (Welbank, 1983). Clearly, as we have suggested, different techniques will favour different kinds of knowledge, both within and between domains.

In this study we have examined plausible combinations of two techniques. In fact we have chosen combinations of a procedural with a declarative technique. This allows us to examine not only the efficacy of individual techniques, but also their efficacies in combination.

We have also restricted ourselves to elicitation directed toward a single target representation - production rules. Expert systems can embody a range of representational formalisms, and production systems are a very popular device (Waterman 1985). It is important to point out that Expert Systems can also be built round structured objects such as semantic nets or frames. The results presented here may not transfer to these alternative representations. We have chosen production rules for three reasons. They are the most commonly used representation. Production rules have a simple and regular structure, this has the added advantage that it is easier to take measures of the quality of the knowledge bases built out of productions. Finally, a purely pragmatic reason, choosing one target representation kept the size of our experimental design under control.

---

(*) Burton and Shadbolt (in press) give a full specification of the techniques.

## The domain

The domain for this study is the identification of igneous rocks. In chosing a domain for which there are many experts and for which an experimental study is possible, we need to restrict it in such a way that the study is feasible, but the domain remains non-trivial. In consultation with an expert Geologist (see acknowledgement at the start of this paper) we constructed a set of 16 igneous rocks which are not trivially discriminable, but which form a homogeneous set. The students used as subjects have learned these rocks early in their University careers and subsequently used, and built upon, these discriminations. For the benefit of anyone reading this article with a knowledge of igneous rocks, the set of rocks used was: Adamellite, Granite, Granodiorite, Kentallenite, Diorite-Tonalite, Nepheline-Syenite, Gabbro, Peridotite, Dunite-Peridotite, Dolerite, Microgranite, Dacite, Rhyolite, Trachyte, Andesite and Basalt. These rocks can be discriminated on the basis of such characteristics as colour, and the crystal shape, cleavage, hardness and optical interference colours of their constituent minerals.

## Method

### Subjects

Subjects were 32 undergraduate Geology students. The students were all volunteers from the 2nd or 3rd year of study, and so had all used the skills necessary in this task repeatedly (see above). We therefore assert that these subjects function as models for genuine experts. Subjects were paid a small sum ( 4) for their participation.

### Design

Each subject took part in elicitation sessions for two KE techniques. Subjects were randomly assigned to one of four groups: interview & laddered grid; interview & card sort; protocol analysis & laddered grid; protocol analysis & card sort. This means that we can compare the four groups

against each other (8 subjects per group) but can also compare the two procedural techniques and the two declarative techniques (16 subjects per group in each comparison). Order of testing in the four groups was counterbalanced so that order could be analysed as a factor.

Dependent measures in this type of study are not easy to establish - what constitutes the efficacy of a technique? Different knowledge engineers will have different constraints on the system development. For this reason we took a variety of measures here, with the notion that engineers can decide for themselves upon the relative importance of the various factors. Dependent measures taken were: time taken for the elicitation session; time taken to code the transcripts of the session into a set of rules; number of rules elicited; number of clauses constituting the elicited rules; completeness of rule set. The final three measures need some explanation.

The total number of rules is simply the number of rules elicited, whether 'correct' or not. As rules can have multiple LHSs, the number of rules is not a good indicator of how much information is encapsulated in the rule set. For example, a typical rule in this domain has the following form:

```
IF  grain size        IS coarse
AND p_clase feldspar  IS absent
AND colour            IS melanocratic
AND content           IS mainly olivine
THEN rock             IS peridotite
```

Therefore we have taken the 'number of clauses' measure; this is simply a count of the LHS elements of all the elicited rules. So, a rule of the form: IF A and B and C and D then E, counts as having four clauses. In addition to quantity, we need a measure of quality of our elicited rule bases. In order to measure this, a senior expert Geologist was recruited to construct a 'gold standard' rule set. The 'completeness of rule set' measure is a comparison of elicited rule sets with this gold standard. That is, how much of the information encapsulated in the gold standard, is encapsulated in the elicited rule base.

In addition to these measures, each subject was given two psychometric tests - personality and cognitive style. It may be that individual experts' characteristics make them more or less amen-

able to the different KE techniques. The personality test constituted the Introvert-Extravert questions from the Eysenck Personality Inventory, or EPI. The cognitive style measure is a psychological differentiation between field-dependent and field-independent subjects (see Witkin and Oltman, 1967). It has been known for some time that people have different 'styles' for thinking. Field-dependent people have a tendency to see problems as a whole, and rely on the context of a given problem. On the other hand field-independent subjects can extract the important parts of a problem from the prevailing context. This dimension is sometimes called the global-articulated dimension and has been used to predict a number of other interesting variables - for instance the styles are differentially prevalent in different professional occupations. This measure was made on the Embedded Figures Test, or EFT (Witkin, Oltman, Raskin and Karp, 1971).

**Procedure**

Each subject was asked to attend two elicitation sessions. Subjects were told the nature of the study, and the set of rocks which we were interested in was explained. Subjects were also briefed on the nature of the target representation. They were shown a production rule from another domain in order that they understood the nature of the rules we were seeking.

The procedure for the formal interview was as described above. For the protocol analysis sessions, subjects were given a randomly-chosen subset of the rocks. A sample of each of the rocks was available, and subjects were asked to make their classification, verbalising wherever possible. A microscope, prepared slides for each of the samples, and other laboratory equipment was available to them.

The card sort was performed with a deck of cards, each bearing the name of one of the rocks in the domain. The laddered grid technique required only a large sheet of paper and a pencil to draw out the domain.

All sessions were recorded, as asides often form useful snippets for knowledge engineers. Any information gathered from one of these asides was counted as coming from the technique being used when it was made. This is a defensible line, as we remain agnostic about whether the techniques work in the simple way they are assumed to work.

**Results**

With regards the overall efficacy of the techniques, two sets of analyses are appropriate, one comparing the four groups, and one comparing individual techniques. This is because the analysis by group represents the combined DVs, and not just the mean of each technique. So, in the combined analysis, completeness of rule set (for example) reflects the completeness of the rule set gained from a combination of both techniques. There is overlap between the techniques, and so the analysis by individual technique provides a different measure - completeness from a single technique

Table 1 shows the results of the analysis of the four different groups. For none of the five measures is there a significant effect of KE combination. Although there are some interesting trends, there is just too small a number of subjects to pick up any but the largest of effects here. The 'number of rules' measure is omitted here for reasons that will are apparent in Table 3. Notably, the card sort techniques reveals very few rules in the first instance, simply a categorical analysis reflecting the number of sorts. This is not really very interesting, and has no bearing on the amount of information present.

TABLE 1

Table 2 shows the results of the analysis of the procedural techniques. The two significant results here are total time ($F(1,30)=4.2$, $p<0.05$) and completeness of rule set ($F(1,30)=39.3$, $p<0.05$); protocol analysis taking longer and giving less information than the interview.

TABLE 2

Table 3 shows the results of the analysis of the declarative techniques. The two significant results here are number of rules ($F(1,30)=46.4$, $p<0.05$) and number of clauses ($F(1,30)=10.7$, $p<0.05$). The number of rules elicited from each technique is uninteresting for the reasons given above. However, it is interesting to note that even though the card sort provides fewer clauses, it still produces the same coverage of the gold standard rule set as the laddered grid.

TABLE 3

We now consider the data from the psychometric measures. Table 4 shows the correlation coefficients relating scores on the EPI to the various dependent measures. This is interesting in that the measure only seems to covary with success in the interview technique. In brief, introverts take longer to complete the interview, but provide more rules and clauses. We shall come back to this in the discussion section.

TABLE 4

Table 5 shows the correlation coefficients relating scores on the EFT to the various dependent measures. Once again, a fairly consistent pattern emerges. Cognitive style only covaries with measures for the protocol analysis and laddered grid techniques. In the case of the laddered grid, field-dependent (global) subjects provide transcripts which are harder to convert into rules than do their field-independent (articulated) colleagues. In the case of the protocol analysis, global subjects provide more rules and clauses than their articulated colleagues, though this does not have a bearing on completeness of rule set.

TABLE 5

We now consider the proportion of coverage given by the various techniques. It seems somewhat surprising from tables 1, 2 and 3 that so little of the gold standard is covered by each of the subjects. In fact almost all the rules of the gold standard are covered by the union of all the subjects, but the different techniques do not seem to access the same proportion. Table 6 shows a break down of the proportion of gold standard rule set by technique. We can see that once again protocol analysis comes out worse than the other techniques.

TABLE 6

Finally, we note that in none of the appropriate comparisons did order have an effect. That is to say, no technique fared better or worse according to whether it was administered first or second. This result holds for all dependent measures.

**Discussion**

One of the most interesting aspects of these results is the poor showing of protocol analysis. Not only does protocol analysis take longer to perform and analyse than the comparable technique (table 2), but it also seems to retrieve a substantially smaller amount of the necessary information than the other techniques (table 6). Of course, our method of applying protocol analysis may be at fault - only a random subset of the rocks was given to each subject. This was done in order to make the time spent in each elicitation session roughly comparable. However, if we should have given more rocks, we can say that protocol analysis is insufferably slow, if we gave about the right number of rocks, we can say that it doesn't provide us with much information. Either way, protocol analysis comes out quite poorly.

The data from the psychometric measures is particularly interesting. Of course when one is

looking at a large set of correlation coefficients, some are bound to come out significant as the type I error rate is going to be quite high. For this reason we have illustrated the level of significance for each of these correlations (tables 4 and 5). The pattern of significance is very consistent in these results. It is perhaps not surprising that introverts take longer on the interview than do extraverts (table 4). However, it seems surprising that they should produce more rules and clauses than the extraverts in order to convey the same amount of information. The cognitive style results are also interesting. It seems that global individuals have some difficulty with the laddered grid. This technique is highly spatial, and so it may be that global individuals find it hard to disambiguate the information contained in the drawing which is made as they progress. This is reflected in the fact the global subjects take longer on the laddered grid than their articulated colleagues. A more difficult result to explain is the fact that global individuals produce more rules and clauses on the protocol analysis than do articulated subjects. If this domain is best conceptualised in 'spatial' terms, perhaps the global subjects find it difficult to conceptualise the problem as they do it, and hence make more of meal of it (these subjects also show a (non-significant) trend towards providing a smaller amount of the domain knowledge than their articulated colleagues).

What can we say about the process of knowledge elicitation from these results? Firstly there are clear differences between these techniques in this domain. Whether this will be the case for other domains will be the focus of future studies. Secondly, there are [illegible] in the efficacy of the techniques across experts. This is an issue which must be faced by future work on KE techniques - it's no good recommending a technique without being sure about who it will work for.

We are now forced to ask whether the different techniques used here are actually tapping different subsets of the necessary knowledge. The results from table 6 show that most of the appropriate knowledge is tapped by each of the techniques. Study of the different rule sets shows that there is no clear dissociation between the knowledge which each of the techniques taps. This leads us to suspect that the techniques do not divide neatly into procedural and declarative as was previously suggested.

A more general finding to emerge from this and early pilot studies (Schweickert et al, in press) concerns an expert's own estimation of the worth of a technique. The impression people gain about the value of an elicitation session bears little relation to the objective information obtained from an analysis of the session. In particular, techniques which force the expert's knowledge into an unfamiliar format, such as card sorts or laddered grids, are much more useful than experts think.

Much remains to be done. We are planning to conduct similar experiments on three other domains, and hope to establish the generality (or otherwise) of our results across domains. We hope that new interactions will emerge as we increase the number of domains and subjects tested.

We are also aware of the limitations of our study. One important restriction is that we have considered only a 'single-pass' elicitation. It is widely accepted that one does not build an Expert System in a simple linear manner: first eliciting the knowledge, and then implementing it. After a first pass elicitation, the practice is to implement a rough and ready version of the Expert System. A knowledge engineer then compares the behaviour of the system with that of the expert. It is usual to carry out a second pass elicitation in which the expert is interviewed in front of the prototype system. This stage often reveals a great deal of extra information, as well as allowing fine tuning of the system.

In fact Expert Systems are usually subject to many stages of refinement - and we refer to this as multi-phase elicitation. Part of our research can be seen as an attempt to pin down more efficient KE techniques in order to reduce the number of passes necessary in development. One direction for future research is to instate a form of structured second phase elicitation as another type of technique. It would then be possible to investigate how much information, and of what sort, is obtained to supplement the first pass knowledge elicitation.

**References**

Anderson, J R *The Architecture of Cognition.* Cambridge, Mass.: Harvard University Press, 1983.

Burton, M and Shadbolt, N R *Knowledge Engineering* in "Expert Systems for Users" Eds. N Williams and P Holt, McGraw-Hill, In press. Also available as TR 87-2-1, Department of Psychology, University of Nottingham.

Gammack, J G and Young, R M. Psychological Techniques for Eliciting Expert Knowledge. In M A Bramer (Ed) *Research and Development in Expert Systems* Cambridge: CUP, 1985.

Grover, M D. A Pragmatic Knowledge Acquisition Methodology *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* 1983.

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

Ryle, G. *The Concept of Mind* Hutchinson, 1949

Schweickert, R, Burton, A M, Taylor, N, Corlett, E N, Shadbolt, N R, and Hedgecock, A. Comparing Knowledge Elicitation Techniques: A Case Study *Artificial Intelligence Review,* In press.

Smith, R G and Baker, J D. The Dipmeter Advisory System: A Case Study in Commercial Expert System Development. *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* 1983.

Waterman, D A. *A Guide to Expert Systems.* Reading, Mass: Addison-Wesley, 1985.

Welbank, M A. A Review of Knowledge Acquisition Techniques for Expert Systems. British Telecom Research, Martlesham Heath, 1983.

Witkin, H A and Oltman, P K. Cognitive Style *International Journal of Neurology,* vol. 6, pp 119-137, 1967.

Witkin, H A, Oltman, P K, Raskin, E and Karp, S A. *A Manual for the Embedded Figures Test.* Consulting Psychologists Press, Palo Alto, 1971.

Table 1: Mean DV scores on the four combinations of techniques (n=8/gp)

| | Interview & Ladd. Grid | Interview & Card Sort | Prot. An. & Ladd. Grid | Prot An & Card Sort |
|---|---|---|---|---|
| Elic. Time (min) | 51 | 44 | 61 | 57 |
| Tans Time (min) | 109 | 97 | 129 | 129 |
| Total Time (min) | 160 | 140 | 191 | 185 |
| Total clauses | 170 | 130 | 166 | 143 |
| Completeness (%) | 34 | 39 | 30 | 33 |

Table 2: Mean DV scores for the procedural techniques (n=16/gp)

| | Interview | Protocol Analysis | |
|---|---|---|---|
| Elic. Time (min) | 24.4 | 30.6 | |
| Tans Time (min) | 56.4 | 79.7 | |
| Total Time (min) | 80.8 | 110.3 | ($p < 0.05$) |
| Number of Rules | 54.9 | 45.1 | |
| Total clauses | 94.4 | 75.8 | |
| Completeness (%) | 27.9 | 7.9 | ($p < 0.05$) |

Table 3: Mean DV scores for the declarative techniques (n=16/gp)

| | Laddered Grid | Card Sort | |
|---|---|---|---|
| Elic. Time (min) | 28.5 | 22.7 | |
| Tans Time (min) | 51.3 | 44.8 | |
| Total Time (min) | 79.8 | 67.0 | |
| Number of Rules | 73.9 | 14.9 | ($p < 0.05$) |
| Total clauses | 101.4 | 63.4 | ($p < 0.05$) |
| Completeness (%) | 28.1 | 30.0 | |

Table 4: Correlation coeffs for the EPI with each DV, across the techniques (Introvert = low; Extrovert = high; n=16)

| | Interview | Protocol Analysis | Card Sort | Laddered Grid |
|---|---|---|---|---|
| Elic. Time | -.73** | .21 | .02 | -.53* |
| Tans Time | -.59* | .06 | .20 | -.25 |
| Total Time | -.65** | .11 | .14 | -.36 |
| Number of Rules | -.73** | .43 | .12 | -.07 |
| Total clauses | -.71** | .15 | -.15 | -.17 |
| Completeness (%) | -.43 | -.05 | -.01 | .30 |

* $p < 0.05$; ** $p < 0.01$

-Table 5: Correlation coeffs for the EFT with each DV across the techniques (articulated = low; global = high; n=16)

| | Interview | Protocol Analysis | Card Sort | Laddered Grid |
|---|---|---|---|---|
| Elic. Time | .27 | .46 | .29 | .48 |
| Tans Time | .26 | .31 | -.17 | .62* |
| Total Time | .27 | .39 | .01 | .63** |
| Number of Rules | .39 | .72** | -.01 | .32 |
| Total clauses | .33 | .66** | .13 | .26 |
| Completeness (%) | .43 | -.48 | -.49 | -.02 |

* $p < 0.05$; ** $p < 0.01$

Table 6: Proportion of Gold Standard rule set covered by union of subjects

| Technique | Amount covered (%) |
|---|---|
| Interview | 79.5 |
| Protocol Analysis | 56.8 |
| Laddered Grid | 85.2 |
| Card Sort | 81.8 |
| All (union) | 91 |

# POMESS: A People Orientated Methodology for Expert System Specification

Dr. Dan Diaper

Section of Psychology, Department of Combined Studies,
Liverpool Polytechnic, C.F.Mott Campus, Liverpool Road,
Prescot, Merseyside L34 1NP, United Kingdom.

A people orientated expert system design methodology is described. The method goes beyond User Centred Design in that individuals other than users have a major input to the design method, as does the client organisation for whom the system is designed. The method thus makes explicit many of the implicit aspects of current, more ad hoc, expert system design approaches. The method has been designed with practical implementation in mind and is intended to be no more time consuming or expensive than present approaches.

## Introduction

At present, it is generally admitted that expert systems are hand-crafted, usually for a single application and often for a specific organisation. At the heart of the current approach is a knowledge engineer who mixes expertise in psychology and computer science so as to extract knowledge from a domain expert and then represents it in an expert system. That this is very much seat-of-the-pants engineering reflects the comparative recentness of commercially available expert systems and the rarity of such interdisciplinary expertise. Furthermore, as in non-AI computing before, the computer science engineering has dominated the domain such that far more attention, and money, has been devoted to the structure and algorithms of the machines than to the psychological issues of interface design and knowledge harvesting. This has occured even though it is widely recognised that "knowledge acquisition is a bottleneck in the construction of expert systems" (Hayes-Roth et al, 1983; Shaw and Gaines, 1986) and that expert systems are operationally defined as machines that have "the concrete objective of simulating an expert's behaviour" (Zarri, 1984). The knowledge engineer tends to rely on the verbal reports of enthusiastic domain experts and a technique often called "rapid prototyping" which is really no more than simple trial and error. It is perhaps not surprising that even less consideration appears to be given to the users of expert

systems than in traditional computer science applications. In fact, some researchers still believe that interface issues should only be considered at the end of system development (e.g. Oliver, 1986, 1987a & b). However, even those computer scientists who congratulate themselves in adopting "User Centred System Design" (Norman and Draper, 1986) seem to fail to appreciate that this is not adequate. First, people are poor at providing verbal descriptions (Nisbett and Wilson, 1977; Bainbridge, 1979; 1986). Second, system users are not likely to be in possession of all the information necessary for system design. In particular, users are likely to be unaware of their organisation's requirements which are rarely the same as their personal ones. The requirements of an organisation, as well as those of the individual, have been poorly analysed in traditional computer design methods such as LBMS or Jackson Structured Design (Cameron, 1984) and are at least as poorly represented in expert system design.

POMESS tackles these issues directly. It starts with a thorough analysis of the stucture of the client organisation. It identifies and analyses the tasks of the potential direct users of a new system and also the other individuals within the organisation who will be influenced by the new system even though they do not use it (indirect users). The domain expert's tasks are analysed in conjunction with the analysis of the users and is ideally done in a non-disruptive fashion during the client orgnisation's normal operations. Diaper (1986b) has already pointed out that task analysis is of limited utility early in the design process, particularly for the design of new systems, as it requires extant tasks that can be analysed. POMESS incorporates a method that in essence simulates a new system before it exists, thus overcoming this limitation of task analysis. The simulation involves using a person, unknown to the user, in place of the system to be designed so that tasks are now available for analysis. This method is known as the "Wizard of Oz" technique and has been used in the past to simulate natural language processing interfaces (e.g. Small and Weldon, 1983), for command-line interface design (Good et al, 1984) and the author has developed the technique for the identification of the knowledge necessary in intelligent interfaces to expert systems (Diaper, 1986a). In POMESS the Wizard of Oz simulation technique employs the domain expert as the simulated expert system. Following a suggestion of Coombs and Alty (1984), the real expert system is built off-line and progressively replaces the domain expert. Contrary to the author's usual beliefs, interface design is done relatively late in the POMESS method. This is because at the simulation stage an extremely powerful interface is supported which provides a rich source of data for the design of a more limited interface which will, however, be sufficient to support the likely user-expert system dialogues. The organisational and task analyses are used to evaluate the final delivery system in its organisational environment and the impact of the new system, which should have been predicted prior to delivery, should be assessed on the

different types of user and on the organisation itself.

The POMESS method can be loosely divided into six stages:

1. Organisational Modelling
2. Identification of People and Tasks
3. Task Analysis
4. Wizard of Oz Knowledge Harvesting
5. Expert System Prototyping and Interface Design
6. System Evaluation

There is some overlap, particularly at stages 2 and 3 which would normally be carried out concurrently. Furthermore, most of the prototyping of the new expert system is done during stage 4 in conjunction with knowledge harvesting. Each of the six stages will be briefly described in turn. The background rationale is not explored any further in this paper but is elaborated in a sister paper (Diaper, 1987).

STAGE 1. Organisational Modelling

In POMESS an organisation may be a self-evidently complete organisation such as a company or institute or it may be only a particular section of a larger organisation. In the real world it is often difficult to determine the size of an organisation as, for perfectly good reasons, many organisations are for some purposes treated as independent entities and for others are part of a larger, parent corporation. The organisational model first constructed in POMESS concentrates on three separate issues:

(i) the interaction of the organistion with the outside world
(ii) the internal structure of the organisation
(iii) the tasks carried out within the organisation

POMESS thus recognises the possible boundary problems in determining the definition of an organisation as it pays attention to both inter- and extra-organisational issues and where possible, would use the same analysis techniques on either side of the boundary. However, particularly where the boundary between the client organisation and the outside world is clear, it is often the case that access to the extra-organisational world is severely restricted and much of the data available will be based on the organisation's current dealings with the outside world and on second-hand, public information (e.g. government policy statements, newspaper reports of the economy etc.). One interpretation of many of the apocalyptic tales associated with the introduction of information technology in commerce and industry is that the internal efficiency of an organisation was raised at the expense of efficiency in dealing with the outside world. At a minimum it must be desirable to specify the client organisation's inputs and outputs and to have at least a reasonable notion of how the external world will

respond to any changes.

Assuming that most of the data will be only collectable from within the client organisation then this paper will concentrate on this analysis. Organisational models within POMESS are represented as a network where the nodes are either individual people or, where individuals have more than one job (e.g. the shop-floor worker who is also a union representative), then each role is represented by a separate node. The important data is not, however, the nodes but the flow of power between them. Power within the POMESS organisational model is defined operationlly as one of the following:

(i) goods
(ii) services
(iii) responsibility
(iv) money
(v) information

Thus POMESS provides a richer model than is usual as normally only one or two of these properties are modelled. Data is initially collected from informal and semi-structured interviews though this is not adequate if it is the only data source used. More detailed, and time consuming, analyses may be targeted by the use of the first pass interview data to identify where there appears to be a lack of reciprocity of the flow of power between the model's nodes. It is assumed that in established organisations that such reciprocity is essential for the maintainance of stability. Another indicator of weakness from the interview data is where different interviewees provide conflicting information. There are a number of non-interview methods that can be used to resolve these initial weaknesses in the model. Two such techniques are:

(i) forced choice bi-polar constructs
(ii) choice dilemma

The former is based on Kelly's Repertory Grid technique (Kelly, 1955; Fransella and Bannister, 1977) where individuals are required to generate constructs that differentiate alternatives. The advantage of this technique is that the constructs are those of the interviewee rather than having been supplied by the interviewer. Choice dilemma techniques present people with critical scenarios that must be solved and which can be used to magnify the differences in reporting found in the interviews.

The first use of the organisational model is in identifying tasks within the organisation that may be suitable for automation and the people who:

(i) will be directly involved running the expert system to be designed
(ii) will be indirectly affected by the new system
(iii) the domain expert(s) whose expertise is to be captured

The organisational model will be used at a number of later stages in POMESS. In particular it should aid the design of the final delivery system and be capable of predicting both the impact of the new system on the organisation and on all the individuals involved. However, the organisational model developed in this first stage has been only based on verbal reports, no matter how sophisticatedly obtained, and not on observed behaviour. Subsequent stages in POMESS place a great deal of emphasis on non-verbal behaviour and it is likely that the organisational model itself will require some modification as such data becomes available.

STAGE 2. Identifying People and Tasks

There are three classes of people who need to be identified following the initial organisational modelling stage and in conjunction with TA. These are:

(i) direct potential users of a system
(ii) indirect users
(iii) domain experts

Given the proper functional definition of expert systems that they emulate some aspects of the behaviour of a human expert (Diaper, 1984; Green et al, 1987) then at least one domain expert is required for building an expert system and there have been moves recently to use a number of experts (e.g. Shaw & Gaines, 1986). The description of the POMESS methodology is restricted to the use of single domain experts in this paper, though this is only for ease of description. In contrast to other expert system design approaches, the important people to identify within POMESS are the users. There is little point in building any system if it is not going to be used or is not usable. The ideal starting point for developing an expert system will be the case where a domain expert(s) has been identified who performs a number of tasks that are suitable for automation and which fulfil the criteria that these tasks are :

(i) relatively common (and thus financially justify subsequent work)
(ii) require the special and rare expertise of the domain expert
(iii) are all within a restricted domain

The latter point is important in that an organisation's general "trouble shooter", who deals with a great diversity of problems, probably uses forms of procedural knowledge that are not well captured in the relatively static knowledge

structures of current expert systems. Such generalists, one suspects, have acquired the skill of rapidly acquiring the domain knowledge sufficient for their purposes as each new and different task is presented. In addition to such "traditional" conditions it is also clearly necessary that:

(iv) the domain expert is regularly consulted for her/his expertise (i.e. by the potential expert system users)

The method of task analysis prefered in POMESS is described in the next section. This analysis will, rather traditionally, concentrate on the two types of people directly involved in the tasks of interest: (i) the domain experts; and (ii) those who consult the expert (i.e. the potential users). However, POMESS also adds an additional aspect to this task analysis. The history of each task should also be investigated. Thus once a potential user has visited the domain expert some effort should then be made to discover the reasons for the inquiry, the pre-conditions, those individuals who would use or be affected by the response (the indirect users) etc. The possession of an organisational model will directly facilitate this sort of investigation as the flow of information between the relevant people, for example, has already been established in stage 1. Modifications to the organisational model may also be necessary, particularly in adding detail to those parts of the organisation that will be most immediately affected by the proposed expert system. Thus the analysed tasks are fitted into their organisational context, rather than simply existing in isolation. It is also possible that the relationships between tasks will become evident or even that indirect users will be so frequently involved in the sorts of tasks that the proposed expert system will deal with that they might profit from being made direct users of the new system.

STAGE 3. Task Analysis

The method of task analysis preferred in POMESS is Task Analysis for Knowledge Descriptions (TAKD) and was developed by the author firstly for the design of syllabi (Diaper & Johnson (in press), Johnson et al, 1984a, 1984b) and subsequently for system design (Johnson et al, 1985). Like all methods of task analysis TAKD starts with the detailed observation of a task and the production of a simple list of the activities involved and the states obtained (an activity list). In POMESS these task analyses centre on the individuals identified in the previous stage, particularly the potential direct users and the domain expert. Ideally, the observations involved are made during the course of the domain expert's normal working day. The task observer should adopt as unobtrusive a role as possible. Experience suggests that video recordings are less useful than the naive might expect and are often felt by those being observed to be

distracting and highly intrusive. The author has found that a well trained observer can, in most cases, make perfectly adequate notes and that a simple audio recording can provide sufficient further detail in the few instances where such notes turn out to be inadequate. As the domain expert is usually involved with communicating with the other person verbally then this, of course, prevents a concurrent verbal protocol. Walk through techniques can, however, be employed after a task to allow the domain expert and also, if desired, the other person, to provide a commentary on the task and a high quality audio recording can be very useful for this.

The advantage of TAKD over its rivals (see Wilson et al (1986) for a review) is that it provides high level generic descriptions of the components of tasks that are hardware independent. TAKD achieves this by a method of generalising a list of the individual actions of a task participant (Specific Actions) and a list of the objects used in the actions (Specific Objects). Tasks can be redescribed using the generalised forms of these lists (Generic Actions and Generic Objects) which are combined as the lexical items in a Knowledge Representation Grammar (KRG). The KRG sentences that describe task steps are independent of the fine implementation details of individual tasks and so allow the commonalities that exist across different tasks to be identified. TAKD could, for example, inform the analyst of the common and distinguishing features of consultation sessions that were face-to-face versus by telephone. Importantly TAKD will also be informative about the relative frequency of similar operations both within and across tasks. The product of TAKD is thus a description of the common elements of the tasks that are likely to be encapsulated in the expert system and at least an indication of their relative importance (in terms of frequency). TAKD can also generate a further description, that of generic tasks by an investigation of the conditional probabilities of sequences or clusters of KRG sentences. For most purposes, this latter analysis is not necessary.

It needs to be stressed that the KRG sentences of TAKD are not themselves an expert system implementable knowledge representation. They provide, however, an invaluable representation of how a wide range of relevant tasks are carried out prior to automation and these can be used directly as a minimum requirements specification when evaluating a new system (because the use of the new system can be described by the same KRG sentences as found in the initial analyses). They also provide a major input to the design of the knowledge harvesting simulations employed by POMESS and described in the next section.

STAGE 4. Knowledge Harvesting

The Wizard of Oz technique has been explicitly developed by Diaper (1986a, 1986b) for overcoming the problem that TA needs extant tasks to analyse. The technique involves the simulation of an unbuilt system by using a person to act as if they were the system (the system subject). Again, as in the TA stage, POMESS allows the minimum disruption to an organisation during the knowedge harvesting phase since if possible the simulation of the expert system will be carried out on-site. For the sake of simplicity, it will be assumed in the description below that a user would normally visit the domain expert personally. Simulations are also possible by telephone if the organisation was prepared to invest in speech I/O technology, though this is not yet at a stage where this is commercially feasible for most expert system applications.

The basic situation requires that rather than a user meeting the domain expert in her/his office they use a terminal (either on their desk or in a separate room) and conduct their dialogue with the domain expert via keyboard and screen. At the other end of the user's terminal is the domain expert and usually a typist. The dialogue between user and domain expert (system subject) is the major data source for knowledge harvesting. A second source of very useful data is the dialogue between the typist and domain expert which can be captured on a tape-recorder, though this data is not essential and not always informative. Knowledge harvesting is done on the user-system subject dialogues in any of the usual ways (e.g. Gammick and Young, 1984), depending on the representational format that will be used in the expert system. The domain expert then identifies the knowledge s/he used by running through the dialogues after each session with the knowledge engineer.

In Diaper's early simulations all the dialogues were constructed on-line with no canned text available. This lead to relatively poor simulations in that many of the user subjects were not convinced that they were dealing with a real system because responses times were not consistent with the users' expectations (i.e. the response time principally depended on the length of a response rather than the difficulty of the question). This was particularly obvious when very long responses were required from the system subject. In Diaper's later work the addition of a small amount of canned text of the sort that machines are typically good at producing (e.g. a list of decisions already made during the interaction and presented to the user subject as a table) greatly enhanced the simulation to such an extent that even two M.Sc. computer science students failed to realise that they were not dealing with a real system.

One of the major functions of the results from TAKD is to specify the sort of interface that will minimally be required for the simulation and the sort of canned text that may be

required. At present all of the author's experience has been with textual interfaces. However, while complex graphics are difficult to provide in simulations, a library of graphics could be available if this was an essential requirement. For example, in an electrical engineering application, a range of prepared circuit diagrams could be available on-line to the domain expert who could have them immeditely modified during the interaction by, for example, placing an arrow on a relevant point, before having the diagram sent to the user subject. The generic descriptions produced by TAKD allow the necessary elements in the interface to be related to their non-automated equivalents. For example, the equivalence of a circuit diagram from a manual and its VDU alternative are easily identified and represented by TAKD.

A second use of TAKD's results is to either ensure that the tasks performed in the simulation are representative of the tasks usually carried out by the domain expert, or, if such real tasks are not feasible, to allow the construction of highly realistic exemplar tasks that can be used in the simulation exercises. This latter route is not prefered but may be the only option available in some circumstances, for example, when the users are not members of the client organisation.

The prototype expert system can be built up as more simulations are performed and once it is capable of some basic functioning the typist can consult the prototype expert system before the domain expert. Simulations can proceed until the domain expert is redundant, at which point a working, prototype expert system will be available. It is likely, however, that there will be a small number of instances where the prototype expert system will fail and that no amount of "bodging", by using special instance rules, for example, will satisfactorily correct these failures. It is essential that the prototype expert system possesses the capability to recognise these instances of failure so that alternative procedures in the organisation can then be adopted without further unsuccessful attempts to use the system (e.g. on recognising such failures call a human expert).

STAGE 5. Interface Design

At this stage of development the prototype expert system is not ready for delivery as the users have never interacted with the expert system except via the human interface (the typist). The dialogues can be re-analysed and the interface designed from them in the same way as the expert system has been designed. Given sufficient technological sophistication, it is possible to design the interface and prototype expert system together. However, this is likely to stretch the processing resources of nearly any system so that in most cases the interface will be better designed after the development of the prototype expert system. Interfaces are in

fact harder to design than expert systems (at least good, intelligent ones are), though users are able to tolerate a wide range of interface styles without great loss of task efficiency which means that bodges in the interface software are more feasible than in the expert system. Thus building the interface after the prototype expert system is prefered because the separation of the two allows a clear identification of where failures occur during development. It is particularly important to recognise unsolvable failure in the expert system as opposed to interface failures, which can usually be overcome (because users are highly flexible). This separation also allows for additional prototyping of the expert system and preparing it for placing in the delivery system.

Diaper's (1986a) results strongly indicate that when full natural language is available in such simulations, only a very small sub-set of the theoretically vast range of facilities are actually required by users. The strategy explicitly suggested is that the user is intially provided with the most powerful interface possible (e.g. a system subject who is allowed to use the full range and expressive power of natural language) and that the dialogue data is used to select desirable features. In theory the interface could be developed in the same way as the expert system: off-line development using the dialogues data and then progressively substituting the interface for the typist. In practice, however, this is not feasible until a substantial amount of the interface has been built. Users cannot be expected to tolerate the high frequency of failure that a typist, who has been instructed in the task, can be expected to deal with, particularly as there is no escape contingency for the user (i.e. the typist is instructed to use the prototype expert system being developed first and when this fails s/he asks the domain expert). The author has been working on the design of an automated analysis technique, called the Wizard's Apprentice, which allows natural language dialogues to be analysed by a cooperation between a human analyst and a computer program. The current, non-intelligent version of the programme has been designed to progressively replace the dialogue analyst as the programme is modified. This is a second and highly unusual form of the Wizard of Oz technique which is still under development. The method of analysis, however, follows that described by Diaper (1986a) though greater detail is possible when the dialogue analyst is supported by even a non-intelligent computer program. At the end of this stage of POMESS, however the dialogues are analysed, the prototype expert system and its interface will have been developed and these can now be placed in the final delivery system.

STAGE 6. Evaluation

Evaluation of any expert system is crucial and the early stages of POMESS strongly facilitate the evaluation process.

First, the expert system can be used, on site, by its direct users and these interactions can be recorded by spooling directly to a hard disc or printer. The users may not even be aware that this data is being collected. Second, the consequences to the indirect users, who have been previously identified, can also be be examined. Third, the consequences for the domain expert can be examined. The evaluation should repeat the initial TAKD exercises and performance can be compared with the results from these initial task analyses. The organisation model can also be used as a bench mark to examine how practices change with the introduction of the expert system. Furthermore, it will take some time before all of the consequences of distributing expertise within an organisation via the introduction of an expert system become apparent. Thus, after an initial evaluation on delivery, it is advisable that follow up evaluations are performed to ensure that the system not only meets the client organisation's requirements and that it is regularly used, but also that there are not negative consequences, or that these are at least minimised.

It also needs recognising that any expert system will not simply replace a human domain expert and the expert system's limitations need to be clearly understood by the client organisation on delivery. Since there will be times when the expert system fails and that this has been recognised during development, the operating procedures of the organisation, and the role of the domain expert, will have to change. Advice about such changes should be part of the package provided from those following the POMESS methodology.

Furthermore, users will require proper training in the use of the system. The early versions of the delivery system might, however, be used by those who have been involved in the simulations as users, which should cut-down initial training over-heads. TAKD was first developed for the specification of training in IT (Diaper and Johnson (in press); Johnson et al, 1984a & b) and thus the evaluation of the system in use and a training programme for new operators may be developed together using this well established application of TAKD.

## In Conclusion

At present, no expert system has been designed using POMESS, though the component parts of POMESS have been investigated in isolation. In fact, any claim to originality that POMESS has must be primarily concerned with the organisation of its components rather than their novelty. The next stage is to test the complete methodology in the design of a system. To encourage industrial or commercial development of POMESS it is obviously important to consider the potential expense of using POMESS. Time and other costs can be considered in two ways: First, as a research programme to develop the first system using POMESS. Second as a "normal" method of expert

system development.

Dealing with the research programme first, there are many aspects of POMESS that will require time and effort but which, once done, may actually be saleable products in their own right. One such example is the simulation software. Diaper currently uses a desk top UNIX system (1Mb. RAM, 20 Mb. hard disc) and the software used for user-system subject communication and data capture runs to several thousand lines of C code. A generic form of this software might be part of a saleable POMESS package. The automation of the analysis of the simulation dialogues, the Wizard's Apprentice program, might also provide part of a POMESS software package. Thus the first development of an expert system using POMESS may take two or three years, depending on the type and structure of the target organisation and the application domain of the expert system.

The development and marketing of POMESS as a complete expert system design methodology, with associated software, could reduce the time for expert system development using POMESS so that it was at least comparable with current development times, if not actually faster. It may well be faster when compared to the usual ad hoc "methods" currently associated with the development of numerous rapid prototypes as POMESS trades off a longer early analysis stages against fewer, later prototypes. Furthermore, POMESS promises to provide a system that satisfies the requirements of individual people and their organisation, something that few other design methods can claim.

## Acknowledgements

The development of POMESS has been undertaken as part of Alvey Project MMI/098 - "The Human-Expert System Interface".

## References

Bainbridge, L. (1979) "Verbal reports as evidence of the process operator's knowledge." International Journal of Man-Machine Studies, 11, 411-436.

Bainbridge, L. (1986) "Asking questions and accessing knowledge." Future Computing Systems, 1, 2, 143-149.

Cameron, J. (1984) "JSP & JSD: The Jackson approach to software development." IEEE Computer Society Press.

Coombs, M, & Alty, J. (1984) "Expert systems: An alternative paradigm." in "Developments in Expert Systems." (Ed.) Coombs, M. 135-158. Academic Press.

Diaper, D. (1984) "An approach to IKBS development based on a review of 'Conceptual Structures: Information Processing in Mind and Machine' by J. F. Sowa." Behaviour and Information Technology, 3, 3. 249-255.

Diaper, D. (1986a) "Identifying the knowledge requirements of an expert system's natural language processing interface." in "People and Computers: designing for usability." (Eds. Harrison, M.D. and Monk, A.F.), 263-280. Cambridge University Press.

Diaper, D. (1986b) "Will expert systems be safe?" in "Second International Expert Systems Conference." 561-572. Learned Information: Oxford.

Diaper, D. (1987) "Designing systems for people: Beyond user centred design." in "Software Engineering" Proceedings of the SEAS Aniversary Meeting Conference.

Diaper, D. and Johnson, P. (in press) "Task analysis for knowledge descriptions: Theory and application in training." In "Cognitive Ergonomics." (Eds. Long, J. and Whitefield. A.)

Fransella, F. and Bannister, D. (1977) "A manual for repertory grid technique." Academic Press.

Gammack, J. and Young, R. (1984) "Psychological techniques for eliciting expert knowledge." In "Research and development in expert systems." (Ed. Bramer, M.) 105-112. Cambridge University Press.

Good, M., Whiteside, J., Wixon, D. and Jones, S. (1984) "Building a user-derived interface." Communications of the ACM, 27, 10, 1032-1043.

Green, D., Colbert, M and Long, J. (1987) "Towards a transaction approach to knowledge-based system development." in "Third International Expert Systems Conference." 207-214. Learned Information: Oxford.

Hayes-Roth, F., Waterman, D and Lenat, D (1983) "Building Expert Systems". Addison-Wesley: Mass.

Johnson, P., Diaper, D and Long, J. (1984a) "Syllabi for training in information technology." in "Contemporary Ergonomics 1984" (Ed. McGaw, E.) 92-97. Taylor and Francis: London.

Johnson, P., Diaper, D and Long, J. (1984b) "Tasks, skills and knowledge: task analysis for knowledge based descriptions." in "Interact'84" (Ed. Shackel, B.), 1, 23-27.

Johnson, P., Diaper, D and Long, J. (1985) "Task analysis in interactive systems design and evaluation." in "Analysis, design and evaluation on Man-machine systems." (Eds. Johannsen, G., Mancini, G. and Martensson, L.). Pergamon Press: Oxford.

Kelly, G. (1955) "The psychology of personal constructs." Norton:N.Y.

LBMS. "LBMS structured development method." 22 Newman St., London W1.

Nisbett, R. and Wilson, T. (1977) "Telling more than we can know: Verbal reports on mental processes." Psychological Review, 84, 231-259.

Norman, D. and Draper, S. (eds.) (1986) "User Centred System Design." Lawrence Erlbaum Associates.

Oliver, A. (1986) "Control of rapid prototyping in expert system development." in "Second International Expert Systems Conference." 247-252. Learned Information: Oxford.

Oliver, A. (1987a) "How to make rapid prototyping effective when developing expert systems." in "Third International Expert Systems Conference." 45-48. Learned Information: Oxford.

Oliver, A. (1987b) "Techniques for expert system testing and validation." in "Third International Expert Systems Conference." 271-276. Learned Information: Oxford.

Shaw, M. and Gaines, B. (1986) "Interactive elicitation of knowledge from experts." Future Computing Systems, 1, 2, 151-190.

Shpilberg, D., Graham, L. & Schatz, H. (1986) "Expertax: An expert system for corporate tax planning." Second International Expert Systems Conference, 99-124. Learned Information: Oxford-New Jersey.

Small, D. & Weldon, L. (1983) "An experimental study of natural and structured query languages." Human Factors, 25, 253-263.

Wilson, M., Barnard, P. & MacLean, A. (1986) "Task Analysis in Human Computer Interaction." IBM Hursley Human Factors Laboratory Report HF122.

Zarri, G. (1984) "Expert systems and information retrieval: An experiment in the domain of biographical data management." in "Developments in Expert Systems." (Ed.) Coombs, M. 201-220. Academic Press.

# KNOWLEDGE ACQUISITION FOR METHOD AND TOOL SELECTION

H.L. Hausen — H.J. Neusser
GMD, Schloss Birlinghoven, D-5205 Sankt Augustin 1, FRG

August 1987

**Abstract**

Software development is to be automated by an effective selection of appropriate methods and tools. This selection could be made by a rule system for both methods and tools. The knowledge for such rule systems has to be obtained from descriptions of the tools and methods and from expert inquiries. But method and tool knowledge.can also be obtained during the application of the rule systems. The rule application protocols have to be analysed therefore. An additional source are user meetings. For the analysis of the various documents information retrieval techniques are being applied. Methods and tools are abstractly defined by an instrument description scheme.

*Fields:* development methods, tools and techniques.

*Keywords:* development method and tool, knowledge acquisition, rule system.

# Contents


1 Knowledge-based Method and Tool Selection 1

2 Structuring Methods and Tools by Rules 1

3 Different ways of Knowledge Acquisition 2

4 Knowledge Acquisition by Information Retrieval on the Handbook 3

5 Knowledge Acquisition by Protocol Analysis 3

6 Knowledge Acquisition by User Meetings 4

7 Summary 5

A Appendix: Description Scheme for Tools 6

B Appendix: Retrieval of Instrument Information 9

# 1 Knowledge-based Method and Tool Selection

A number of methods and tools are currently in use in order to perform specific software engineering tasks. For most tasks one will find no generally accepted method and so it is necessary to apply combinations of several methods. Since methods can overlap, complement each other or be independent, it is necessary to support the selection of methods. To increase the productivity and to lower the costs effective tools should be available to ease a software engineer's work.

The identification and selection of appropriate methods and tools is also an area where specific instruments might assist an engineer. Method and tool selection may be provided by a method rule system and a tool rule system. Prototypes of such rule systems have been developed for the validation of software as a software engineering task and Unix as a tool environment ([Neus87], [Haus87a] and [Haus87b]).

These systems consist of method-, tool- and interrelation-rules. After a tester has selected a validation method, he is enabled to get information on which tasks of the method are supported by which Unix tools and how these tools have to be applied. It might happen that he gets more than one instrumentation. In this case he might decide his favourite tool. The interrelation-rules represent knowledge on how to decompose such method tasks which are too complex to be instrumented immediately by Unix tools. Since these decompositions are not part of the validation methods the interrelation-rules are used to define these decompositions. It is evident that there exist tasks for which a decomposition does not lead to subtasks being instrumented by Unix or for which a decomposition is not possible at all. Such tasks have to be performed manually.

# 2 Structuring Methods and Tools by Rules

Rules are well suited for representing heuristic knowledge and experiences, a fact which is important in nearly all aspects of software engineering. Generally, rule systems are not deterministic. For our purpose, nondeterminism is necessary since in most cases there is no general accepted method or tool.

Generally, a rule consists of a condition-part and an action-part:

| | |
|---|---|
| **IF** | condition part |
| **THEN** | action part |

Neglecting parameters and weight factors the method rules have the following general form:

| | |
|---|---|
| **IF** | method predicate |
| **THEN** | method predicate |

where the IF-part refines the THEN-part.

To be able to select tools depending on a special task you have to represent knowledge about the applicability of those tools regarded. Knowledge about tools is represented by rules.

Neglecting parameters and weight factors the tool rules have the following general form:

| | |
|---|---|
| **IF** | tool predicate |
| **THEN** | tool predicate |

The selection of tools for a special method task is reached by an inference process. An inference engine is looking for a tool predicate matching the method predicate you wish to instrument. In some cases you will find several tools, in other cases only one or even no tool.

Neglecting parameters and weight factors the interrelating rules have the following general form:

| | |
|---|---|
| **IF** | interrelate predicate |
| **THEN** | method predicate *or* interrelate predicate |

There is no need to have method predicates in the condition part of this rule because methods are defined by the method rules.

On the other hand very often it might be possible to map a (small) method straight forward onto a tool or a set of tools.

| | |
|---|---|
| **IF** | tool predicate |
| **THEN** | method predicate |

But such rules have not to be defined because the matching establishes such relations. The interrelation rules are matched against the tool rules and other interrelation rules, but not against the method rules.

# 3 Different ways of Knowledge Acquisition

In this paper we describe three different ways of knowledge acquisition for method and tool selection:

- analysis of the handbooks
- analysis of usage protocols
- analysis of expert meeting reports

Motivation for this kind of approach is that computer readable information is easily available, namely the manual in files as well as usage protocols. You may even provide that expert meeting reports will be written in a file. This approach may be used to ease the knowledge acquisition when developing rule systems for problem domains for which a method and tool selection is to be provided.

# 4 Knowledge Acquisition by Information Retrieval on the Handbook

Unix is a largely complex operating system with lots of tools (more than 200). As a result of this the system's handbook is of a huge volume and thus difficult to use. The problem even increases in case of heterogenous tool systems, for example Unix versions BSD 4.2, V.2 Ultrix, Aegis, .... In such a case a user has to go through a number of handbooks. For example, in BSD 4.2 there is an option for the Unix command **cat** which provides a numbering of the files handled by **cat** which is not available in version V.2.

In the following we discuss how to increase the effectivity in using the handbook. Since the Unix handbook is available as a set of files it is possible to apply information retrieval methods to find more appropriate tools than currently proposed in the rule system or to decide which tool is best in case that several tools have been proposed.

The idea is to search for selected strings in the whole manual automatically instead of manually. Which strings are appropriate to be searched for?

As a first step classification schemes of methods and tools may be used (cf. appendix A). As an example, the input of software validation means may be classified as follows: text, code or data.

So you may search for the string 'text' if you have a task concerning some text and you will get a list of all commands dealing with text. Thus you get a subset of the Unix tools which is far easier to handle. This may be useful in case that the rule system proposes several tools and you have to select one of them. Searching for some key words you are especially interested in may help to find the most appropriate tool for the task which is to be solved.

The following example shows that in some cases it is not sufficient to search for only one key word. Suppose you are searching for tools which format programs. The first trial will surely be to search for the key word 'format'. In most Unix environments you will not find the command **cb**, though it really formats C-programs. But you will find **cb** when searching for the key word 'indentation'.

# 5 Knowledge Acquisition by Protocol Analysis

A rule system as mentioned in section one provides its users with all methods and tools appliable for their current task. When examining usage protocols of the rule-systems we found that in practice users prefer some method or some tool for a special task. For example, in software testing a user always selected branch testing instead of statement testing, multicondition testing or path testing. On the tool side, in our example a Unix environment, a user always used **grep** as a pattern matcher instead of **egrep**, **fgrep** or **awk**. These observations led us to the idea

to adapt the rule systems to a user's point of view. By seeing only his favourite methods and tools the work with the rule systems is simplified.

This user adaption may be realized by adding user predicates to each rule. As a result of this every user gets his own virtual version of the rule system. Additionly, however, we propose to inform a user about the existence of alternatives to his favourite methods and tools. In the next section this aspect will be discussed more detailed.

Another approach for knowledge acquisition is the comparison of usage protocols which are influenced by the rule system with those which are not. The comparison with usage protocols before the user has ever seen the rule system may indicate missing or inconsistent rules. The comparison with usage protocols after the user has gained practical experience with the rule system may indicate which parts of the rule system have been adapted and which not. The parts not yet adapted should be reworked.

# 6 Knowledge Acquisition by User Meetings

In the previous section usage protocols are discussed only with respect to one user. In the following we compare protocols of several users in order to acquire knowledge about user-oriented method and tool selection. Regarding a special task you may find that user A selects method M1, user B selects method M2 and user C selects method M3. On the tool side, user A may prefer tool T1, while user B prefers tool T2 for the same task. In these cases the question arises which method and which tool is best. One possible solution is to take other protocols into account and to make statistical examinations: the method which most users applied is best. Especially you have to take into account the user's degree of qualification. So it is thinkable that novices in the domain of software testing rely upon statement coverage although it is known that this method alone is insufficient ([Haus85]).

User meetings may help to overcome the difficulties with respect to statistical examinations. Users may tell each other why they prefer some methods or some tools and why they dislike others. Such discussions may lead some users to new aspects. The results and effects of these discussions may be examined by comparing usage protocols before and after a discussion. It may turn out, for example, that a user applys a tool he has never used before. This may indicate that he has been convinced or at least that he has been made eager to know.

Meetings of experts, both on the method and tool side, may be used for a validation of those results gained by an analysis of the handbooks and usage protocols as well as for the identification of new knowledge to be incorporated into the system.

User meetings should be well organized. They should produce statements on knowledge which can be translated into rules. The use of forms may be helpful to concentrate the discussion on methods and tools with respect to the rule systems.

These forms or even the minutes of the user meetings might be processed by the means discussed in the section on retrieval on the handbook.

# 7 Summary

Generally a user knows only a small part of the Unix tool set ([Drap84], [Norm81]). Thus, the potential possibilities of Unix are seldom exploited. A knowledge-based system helps to reduce the discrepancy between the tools offered by Unix and those actually used. Knowledge acquisition remains a task even during the application of knowledge-based systems. We have proposed three different ways of knowledge acquisition, one concerning the system's handbook, the other two concerning the users of the rule-systems. This approach may be used to develop rule systems for problem domains for which a method and tool selection is to be provided.

Knowledge acquisition must continue during the application of such systems. It is shown how users may be supported by information retrieval methods to find more appropriate methods and tools than currently proposed in the rule system. A simple database or an information retrieval system have to be applied to extract knowledge on tools out of the system's handbook. This is effective if one has a system like Unix where the handbook is of a huge volume, and might be even more effective in a heterogeneous environment, where one has to go through a number of such handbooks. This is especially appropriate where the handbook is available as a file or a set of files or as a data base. On the other hand user protocols may help to extract meta knowledge about the applicability of methods and tools.

Using these informations the rules may be restructured. As a result of this it is possible to adapt the rule systems to the user's point of view (every user his own virtual version of the rule systems). User statistics in addition with user meetings and expert inquiries may help to derive the most appropriate methods and tools. This knowledge may be inserted into the rule system. This kind of knowledge acquisition during the application of such rule systems is especially useful in case of several methods and several tool systems.

# References


[Drap84] W. Draper *The Nature of Expertise in Unix* Proceedings of INTERACT '84, FIRST IFIP Conference on Human-Computer Interaction, Elsevier Science Publishers, Amsterdam, Vol. 2, 182–186,(1984)

[Haus81] H. L. Hausen *A Dynamic Self-Tuning and Adaptive Information Retrieval System.* in: J. Herr, Theoretical Issues in Information Retrieval, Proc. Fourth Annual International SIGIR Conference, 15 p., ACM NewYork, 1981

[Haus85] H. L. Hausen *Effektive Software-Qualitätssicherung durch integrierte Techniken der Software-Validation* ? R.Lauber (Ed.), Technische Zuverlässigkeit, Generalthema: Softwarequalität und Systemzuverlässigkeit, VDE-Verlag, Berlin, ISBN 3-8007-1416-7, S. 63-78

[Haus87a] H. L. Hausen, H. J. Neusser *Knowledge Based Method and Tool Selection Using Parameters and Weight Factors* — to be submitted —

[Haus87b] H. L. Hausen, M. Müllerburg, M. Schmidt *Examination, Measurement and Assessment of Software Products and Projects.* in: Proceedings EOQC'87, Munich, June 3.-5., 1987

[Neus87] H. J. Neusser, U. Schwamborn *Wissensbasierte Software-Validation unter Unix* Diplomarbeit, Bonn, 1987

[Norm81] D. A. Norman *The Trouble with Unix* DATAMATION, nov 1981, 139–150 (1981)

[PCS84] PCS *Muniz-documentation* München, 1984

[SUN86] SUN *System V.2 Documentation* Mountain View , CA, 1986

[DEC86] DEC *System V Documentation*

[APOLLO86] APOLLO *System V Documentation*

[APOLLO86] APOLLO *Aegis Documentation*

[AT&T86] AT&T *System V Documentation*


# A Appendix: Description Scheme for Tools

There exist a number of tool indexes used in software tool data bases. Each of them uses special description models to describe the characteristics of software instruments. An instrument is described by an ordered set of attributes, i.e. by the instrument tuple.

| | Software Instrument Tuple | |
|---|---|---|
| Software-Instrument (INS) ::= { | | |
| | <> Input-Description | (IP) |
| | <> Function-Description | (FN) |
| | <> Output-Description | (OP) |
| | <> Phase-Description | (PH) |
| | <> Quality-Criteria-Description | (CR) |
| | <> Means-Description | (MC) |
| | <> Environment-Description | (ER) |
| } | | |

```
INS ::= {
        <> IP   [     Object ( o1, o2, ..., oJ )
                + Control( c1, c2, ..., cK )
                ]
        <> FN   [     Construction
                      << Requirements definition ( r1, r2, ..., rM )
                      # Descriptive definition ( d1, d2, ..., dN )
                      # Imperative definition ( i1, i2, ..., iO ) >>
                + Validation
                      << Inspection ( i1, i2, ..., iP )
                      # Testing ( t1, t2, ..., tQ )
                      # Verification ( v1, v2, ..., vR )
                      # Symbolic Evaluation/Execution ( s1, s2, ..., sS )>>
                + Administration
                      << Planning ( p1, p2, ..., pT )
                      # Project controling ( c1, c2, ..., cU )
                      # Project reporting ( r1, r2, ..., rV )
                      # Product management ( r1, r2, ..., rW ) >>
                ]
        <> OP   [     User output ( u1, u2, ..., uX )
                + Machine output ( m1, m2, ..., mY )
                ]
        <> PH   [     Requirement phases ( r1, r2, ..., rA )
                + Design phases ( d1, d2, ..., dB )
                + Implementation phases ( i1, i2, ..., iC )
                + Application phases ( a1, a2, ..., aD )
                ]
        <> CR         ( c1, c2, ..., cE )
        <> MC         ( m1, m2, ..., mF )
        <> ER   [     Complementary instruments ( c1, ..., cG )
                + Preceding instruments ( p1, ..., pH )
                + Succeeding instruments ( s1, ..., sI )
                ]
        }
```

Remark: Within ( ), the appropiate entries are listed by name. The above gives the general framework for a definition of software instruments. More details about possible entries may be found in [Haus85].

The FN entry describes for which task a method or tool is meant. In Testing, for example, black box testing, open box testing, structure testing, path testing, and others might be selected. Input and output defines the form of objects consumed or produced (e.g. object code). PH defines the phase a tool or method might be

applicable. The CR defines the (quality or productivity) criteria to be achievable by the application of a specific method or tool. MC allows the indication on whether it is a method or tool, but if required it can also be used to differentiate according to the means classification given above. By ER the environment of the instrument described by a tuple is to be defined.

It should be noted that this scheme is applicable to the description of instruments in a general sense. An instrument might be a concrete software tool, like a Unix command, or a selected activity of a user interacting with a software system. In the later case, the instrument description scheme can be used in the usage protocol as a data structure to define the set of activities performed during a session and their inter-relations. If the scheme is used to describe Unix commands it must be noted that the information about the function and the phases are not retrievable from the command descriptions in the manual. This information has to be added by the system administrator. This makes it clear why rule systems are necessary here to define and make reusable human knowledge.

In order to show the application of this scheme we apply it to describe a compiler. Thus we get:

| **COMPILER** | | |
|---|---|---|
| **Input** | **Object** | **Code** |
| | **Control Input** | **Commands** |
| **Function** | **Validation** | |
| | **Inspection** | - |
| | **Testing** | **Debugging** |
| | | **Static Analysis** |
| | | **Dynamic Analysis** |
| | **Verification** | - |
| | **Symbolic Validation** | - |
| **Output** | **User Output** | **Diagnostics** |
| | | **Listings** |
| | | **Tables** |
| | **Machine Output** | **Object Code** |
| **Phase** | **Implementation** | **Software-Program** |
| **Quality Criteria** | | **Correctness** |
| **Means Class** | | **Tool** |
| **Environment** | **Preceding instruments** | **Program editor** |
| | **Succeeding instruments** | **Linkage editor** |

# B Appendix: Retrieval of Instrument Information

It is assumed that instruments, instrument-requests and instrument-proposals are represented and handled as multidimensional vectors in an abstract space. The retrieval of information on instruments might then proceed as follows:

1. A set DT of n1 terms used for instrument indexing, i.e. the instrument-term thesaurus.

2. A set QT of n2 terms used for query indexing, i.e. the instrument-query-term thesaurus.

3. A set DO of n3 instrument descriptions in natural language.

4. A set QO of n4 instrument-queries in natural language.

5. A set BT of n5 trivial words, i.e. the banal-word thesaurus. The BT is used in the construction of the DT. The following should hold:
   DT = QT = {terms in DO and QO} minus BT

6. A transformation FD on the set of instruments DO mapping each instrument onto the abstract vector space D spanned by the terms of the instrument-term thesaurus DT.

7. A transformation FQD on the set of queries QO mapping each query onto the abstract vector space QD spanned by the terms of the instrument-term thesaurus DT.

8. A set P of previously produced or accepted proposals p, where each $p \in P$ is a subset of D, i.e. $p \in D$.

9. A search function FSD on the Cartesian product of D and QD which induces a partial ordering on the set D. FSD takes a query q and produces the partial proposals $ppd \in PP$, depending on the correlation of the query and the instruments in D. PP is the set of proposals produced by FSD.

10. A transformation FC on the complete proposal PP that generates a subset PPC of PP depending on a selection parameter, i.e. the cut-off value. PPC is the proposal the system presents to the user as the answer to the query considered.

A more elaborated retrieval concept, which includes feedback on instrument documents and on instrument queries, could be developed on this basic model. The elaborated version will guide the retrieval of the appropriate instrument documentation by means of additional user input. Such user input might be either

a reformulated instrument query or a set of nearly-acceptable instrument documents. Currently the basic model is applied to ease the retrieval of instrument descriptions and to analyse user protocols, both described in succeeding sections. The detailed description of the information retrieval approach taken has been described in [Haus81].

EXTENDED ABSTRACT

# Knowledge Acquisition For Small Scale Expert Systems From Consulting Experts: A Field Study Approach

Paul Watkins

Daniel E. O'Leary

Graduate School of Business
University of Southern California
Los Angeles, California 90089-1421

213-743-4092

4720490 USC LSA
674803 UNIVSOCAL LSA

July 1987

Comments are Solicited

Please Do Not Quote Without the Author's Permission

# Knowledge Acquisition For Small Scale Expert Systems From Consulting Experts: A Field Study Approach


## Abstract

The paper differentiates between the knowledge acquisition for small scale and large scale systems, and delineates consulting experts from other experts. It then explores an experiment in knowledge acquisition for small scale systems. The preliminary results indicate that domain consulting experts with minimal computer experience choose problems for expert system representation that are decomposable and well-structured and represent those problems using an expert system shell more rapidly and of better quality than those personnel with less domain expertise, even when those other personnel had more computer system experience. We hypothesize that this occurs because the consulting expert has a strong understanding of the domain and is experienced in selecting and developing solutions to such problems. It also is likely that this occurs because the consulting expert is faced with similar questions about the same problem areas on multiple occasions. As a result, he is likely to have some "pre-established" problems and logical solution methodologies established for those problems that would make good expert systems. Thus, in some cases, in order to identify "good" problems for expert system development and to acquire the knowledge for those systems it may be accomplished best by teaching the expert how to use an expert system shell.

EXTENDED ABSTRACT

# Knowledge Acquisition For Small Scale Expert Systems From Consulting Experts: A Field Study Approach

Recently, DuPont announced that they expected to have over 2000 expert systems in place by 1991 (Williamson [1987]). In order to accomplish the development of this many systems in such a short period of time, it is likely that this announcement includes systems for a variety of problems, ranging from small scale, highly structured problems to very large scale and less structured problems. Small scale problems are differentiated from large scale problems by the fact that small scale systems, in most cases, include knowledge from a single expert, whereas large scale systems may include the interlocking expertise from multiple experts.

One of the primary sources of expertise for small scale systems is the "consulting expert." Typically these experts answer frequent questions in a narrow decision making context. In addition, the questions generally can be answered by the expert without consultation with other experts. Thus, these experts are accustomed to solving decomposable.

These experts generally are in high demand and can rarely meet all the requests for their time. Typically, the experts are resident at some central location in order to meet the requests of others throughout the country and world. The high demand

nature of this situation indicates that the use of expert systems to leverage the expertise would be highly cost effective.

## Impact of Type of Problem on Knowledge Acquisition

Some previous developers of complex, large scale systems have found that they need to become domain "near-experts" as part of the development effort in order to effectively acquire knowledge from the domain expert (Lethan and Jacobsen [1987]). This can be quite an expensive proposition. Accordingly, that same approach would not be appropriate for all problems.

For small structured problems it may be more efficient and cost beneficial for the expert to be taught how to use an expert system shell, rather than have an expert in programming and computer techniques act as an intermediary. In addition, because the expert would then know both the domain and the capabilities of expert systems, the expert may provide an increased ability to identify problems for expert system development. This paper discusses a field study to analyze these issues.

## Consulting Experts

Typically, the consulting expert's expertice is highly specialized and very unique. In his role as a consultant he is likely to face the same questions about similar issues from a number of different sources. In addition, the consulting expert, is constantly consulting with new consumers of his specialized information.

Accordingly, the consulting expert is likely to have a number of alternative solutions to meet the needs of different problems and is likely to have structured solutions to meet the demands of the similar questions. In addition, he is likely to be aware of a number of problems that can be solved using using logical analysis of expert systems. This leads to the following hypotheses.

Hypotheses

$H_1$ = Consulting experts can more rapidly identify problems for expert system development than non-consulting experts.

$H_2$ = Consulting experts will produce more comprehensive and "complete" expert systems than non-consulting experts (i.e., higher "quality").

$H_3$ = Consulting experts will produce expert systems more rapidly than non-consulting experts.

Experiment

Thirteen subjects participated in a field study to test the above hypotheses. Four of the subjects were Senior Tax Practice Partners of a large international accounting firm that other professionals in the company in offices throughout the world would consult when they had a tax problem in a specific area. These experts had minimal computer literacy and were not heavy users of computer techniques. The other nine subjects were

Senior Managers, the organizational level immediately below Partners. Of the nine, six were National Office Tax Technicians who specialized in tax and the application of computer technology to tax. For example, they may have developed templates for Lotus spreadsheets. The remaining three were tax specialists rotated through the national office from the local offices. None of the participants had any previous experience in developing expert systems.

The experiment started with an introduction of expert systems to the group. This included the components of expert systems, a detailed discussion of how inference engines worked and the strengths and limitations of expert systems.

Then the subjects were introduced to an expert system shell, EXSYS. Using EXSYS they were asked to solve two "toy" sample problems (the "bartender" problem and the "animal identification" problem--from Winston [1984]) and two realistic tax problems whose solutions were given to the subjects. After this preparation the subjects were asked to develop two systems based on problems of interest to them.

## Findings

Two of the four consulting experts rapidly became bored with the toy problems and sample problems and started work on their own expert systems. These two experts finished two small scale systems each, in one day. The other two consulting experts also were able to develop systems that they could use in their daily

consulting experience. In addition, during the development of one system the expert found that there were some errors in a published interpretation of a very complex portion of the tax law.

Each of the four domain experts were able to select problems that were bounded and well-defined. They developed their applications faster than the other participants. In addition, these experts also were able to order the rules in order to get the inference process correct.

The other nine subjects had more difficulty in identifying systems that would be of use to themselves. Unlike the consulting experts, the other nine subjects did not have readily identified decision problems that would make good expert systems.

They also had difficulty in implementing expert systems. One of the computer literate subjects had difficulty transforming a flowchart into a knowledge base. Other subjects had difficulty getting the inference process correct for the toy problems.

## Limitation

The primary limitation of the study was the sample size. However, the time and accessibility of true consulting experts for such experimental purposes is highly limited. Because of the nature of consulting experts it is not possible to employ the usual experimental subjects--students. Thus, although the sample was only thirteen, we were pleased to have access to the experts for the purposes of this experiment.

We feel that this limitation is overshadowed by the need to understand the nature of the consulting expert and the resulting experimental findings of this study.

## Implications

This experiment indicates that for single expert, small scale problems using consulting experts, it may be more cost beneficial to teach the consulting expert how to use the computing tools then to have a computing expert solicit the knowledge from the expert and put in a system. A residual benefit can be an increased capability to identify problems for expert system development, because of the expert's better understanding of the capabilities of expert systems.

## Selected References

Boeing Computer Services, Proceedings of the Knowledge Acquisition for Knowledge-Based Systems, 1986.

Lethan, H. and Jacobsen, H., "ESKORT -- An Expert System for Auditing VAT Accounts," Expert Systems and their Applications -- Avignon 87 Proceedings, Avignon, France 1987.

Williamson, M., "At DuPont, Expert Systems are Key to AI Implementation," PC WEEK, January 13,1987.

Winston, P., Artificial Intelligence, Addison-Wesley, Reading Ma., 1984.

# Role of Psychological Techniques and Intermediate Representations in Knowledge Elicitation

Richard M. Young & John Gammack

## EXTENDED ABSTRACT

The purely pragmatic approach to knowledge elicitation (KE) involves the elicitor in chatting informally with the Expert and then building some of what is said into a few rules, after which it relies for further progress on having the Expert iteratively comment on and help debug a "prototype" system. This approach suffers from a number of disadvantages (and indeed to describe it as "knowledge engineering" is surely more a piece of wishful thinking than of accurate terminology). The problems have to do with:

(a) The fact that the approach is unsystematic; and

(b) The coding of elicited knowledge directly into rules (or some other target format).

Using only the weak knowledge acquisition techniques associated with informal interviews means that the elicitation is restricted to verbally articulatable knowledge, and is inadequate for tacit or implicit knowledge. Yet experts' skill is often compiled into an automatic version, so that the experts are unable to report on their own procedures and mental processes. Furthermore, there is anyway a need to capture the main entities in the domain and their relationships *before* beginning to code the rules, since otherwise the system becomes incoherent and unmanageable as the number of rules grows. In short, KE restricted to informal interviews is time-consuming, incomplete, and inefficient.

One of the main disadvantages of trying to code the expert knowledge directly into rules is that the representation scheme is chosen before the content of the knowledge is known and tends to drive the coding, instead of choosing a scheme of representation appropriate to the knowledge (Young, 1984). For example, the representation formalism might be driven by the availability of expert system shells rather than by the need to express the expert's knowledge: "Selection of a knowledge representation scheme prior to the acquisition of the knowledge to be represented is putting the cart

before the horse" (Cooke & McDonald, 1986). Moreover, expertise usually consists of a number of different kinds of knowledge each with its own elicitation requirements (Gammack & Young, 1985), and it is "unlikely that a single type of knowledge representation can efficiently capture expertise" (Cooke & McDonald, 1986).

We address each of these classes of deficiency in turn, (a) the lack of systematicity by the use of psychological techniques, and (b) the direct coding into rules by the use of intermediate representations.

## *Use of Psychological Techniques*

A way of overcoming the disadvantages of reliance on informal interviews is to make use of systematic psychological techniques. We report briefly on some work using such techniques, aimed at eliciting the structure of the basic entities in a domain and their relationships (Gammack, 1987a, 1987b). Firstly, a set of items from the domain are subjected to an hierarchical sorting procedure by the expert, yielding information about the major classes in the domain. The expert is then asked to perform the first of the two main elicitation tasks, which is to rate the similarity between pairs of items. The resulting matrix of proximities is analysed in three different ways:

1. Using the ALSCAL program to perform a multi-dimensional scaling (MDS), which yields a two-dimensional map showing the dominant dimensions of the domain.
2. Using single-link cluster analysis, which yields a picture of the main groupings of items.
3. Using Pathfinder analysis (Schvaneveldt, Durso & Deerholt, 1985) to yield a link-weighted network showing pairwise relations in the domain, which complements the MDS by focussing primarily on closely related pairs.

The three representations resulting from these analyses correspond to maps of the domain at the scale of atlas, regional map, and local street plan respectively. Experts are then given the second main elicitation task, the well-known Repertory Grid, which yields a number of attributes that distinguish between items in the domain. The ratings on those attributes can be used to compute a derived proximity matrix, which can in turn be analysed by the same three methods as were the similarity ratings. Finally, the expert is asked to label the relations represented by the links in the Pathfinder network.

The convergence between the (now six) representations can be examined to provide a cross-check on the consistency of the elicitation procedures. The validity of the knowledge elicited can be checked by using it to predict the expert's performance on further, secondary tasks that we refer to as *validation* tasks. Those used so far (Gammack, 1987a) include: having the expert rate items against various scales; free recall of the items by the expert; and playing the game of Twenty Questions in which the expert has to ask questions to identify an item thought of by the elicitor.

This approach has now been applied in three domains: the selection of statistical techniques for analysing commercial survey data; the choice of information source in an industrial research information centre; and the classification of British steam locomotives. The technique is particularly appropriate in the early stages of KE, where there is otherwise little guidance available. It is able to elicit the expert's implicit knowledge, as revealed by their judgements. As just explained, it makes use of converging evidence to validate the resulting conceptual structures and discover the "enduring psychological reality which is stable across different tasks" (Gammack, 1987a). In short, the approach is quick, can be used with either one or multiple experts, includes procedures for validation, and produces visualisable representations.

## *Role of Intermediate Representations*

Among the characteristics of the psychological techniques just outlined are that they produce non-executable representations, intended more for use by human than by computer, and that they produce *more than one* such representation. In this respect, the psychological techniques can be seen as part of a methodology that encourages the use of multiple *intermediate representations* mediating between the human informant and the final expert system. Other examples of intermediate representations (IRs) are: protocol transcripts (in English prose); systemic networks; diagrams of various sorts (e.g. circuit diagrams); "semantic nets", influence diagrams, etc.; 2D or 3D layouts; and so on.

These IRs can be thought of as lying on a spectrum, ranging between human-like representations at one end and computer-like at the other, with information being transformed among them. We certainly do not wish to imply that KE can be viewed as a linear sequence from human to computer, as it clearly involves all sorts of iterations and convergences through the different IRs, but it is worth trying to understand the systematic changes that occur as one moves along the spectrum from one end

to the other. Some of these gradations are:

1. *Loss of completeness*. As we move from human to machine, the represenations lose information. There are all kinds of information which we can elicit from experts, either directly or indirectly, and which we can perfectly well understand and notate, but which are beyond the current state of the art of knowledge representation for a computer (let alone a standard expert system shell).
2. To some extent compensating for the loss of completeness, there is a *gain in self-sufficiency*. The more extremely human-oriented IRs, such as those using English prose, require semantic interpretation by the reader. They "contain knowledge" only by virtue of the semantic (and other) capabilities that the human brings to bear on them. By contrast, IRs at the computer-oriented end of the spectrum are more self-contained, and require less semantic interpretaion on the part of their user. Indeed, by definition, a machine-executable representation requires only syntactic interpretation.
3. The more machine-oriented IRs suffer a loss of *cognitive compatibility* with their users, who might be the expert, the elicitor, or some other reader. In particular, IRs at the human end are intelligible to and accessible to the expert as well as by the elicitor, and can play an important part in the KE by acting as a shared medium of knowledge communication between them.

In summary, using IRs in the process of knowledge elicitation has a number of advantages. First, the IRs serve as a record of the knowledge elicited that is independent of the technology used (such as a particular expert system shell), and so can act as a source for re-implementing a system later using improved technology, without having to repeat the whole elicitation procedure. Second, IRs can act as a medium of knowledge communication shareable by the expert and the elicitor. Third, use of (particularly a variety of) IRs can provide a more complete record of the knowledge than can be captured solely in a machine-interpretable representation. Fourth, and finally, the knowledge recorded in the IRs can be used for purposes other than building a problem-solving expert system, such as for explanation, for tutoring, and as a basis for system documentation.

REFERENCES

Cooke, N. M. & McDonald, J. E. (1986) A formal methodology for acquiring and representing expert knowledge. *Proceedings of the IEEE, 74*(10), 1422-1430.

Gammack, J. G. (1987a) Modelling expert knowledge using cognitively compatible structures. *Proceedings of 3rd International Conference on Expert Systems,* 191-200. London: Learned Information.

Gammack, J. G. (1987b) Different techniques and different aspects on declarative knowledge. In Kidd, A. L. (Ed.) *Knowledge Acquisition for Expert Systems: A Practical Handbook.* Plenum.

Gammack, J. G. & Young, R. M. (1985) Psychological techniques for eliciting expert knowledge. In Bramer, M. A. (Ed.) *Research and Development in Expert Systems,* 105-112. Cambridge University Press.

Schvaneveldt, R. W., Durso, F. T. & Dearholt, D. W. (1985) PATHFINDER: scaling with network structures. Memorandum MCCS-85-9, Computing Research Laboratory, New Mexico State University.

Young, R. M. (1984) Human interface aspects of expert systems. In Fox, J. (Ed.) *Expert Systems: State of the Art Report.* Pergamon Press.

# A DECOMPILATION APPROACH TO KNOWLEDGE ACQUISITION: USING CONCURRENT EXPERT REPORTS AS THE BASIS FOR AUTOMATED KNOWLEDGE MODELING.

**Otto E. Laske, Dr.phil.**
Department of Computer Science
Boston College
Chestnut Hill, Massachusetts 02167, U.S.A.

In this lecture, I elaborate on a design procedure for knowledge-based systems that is based on a three-step algorithm comprising an encoding, an analysis, and a re-synthesis part. The algorithm presupposes as input a concurrent expert report ("protocol"); it yields as final output a compact production system encoding of the empirical knowledge base implicit in the report.

The algorithm is based on the idea of using a production system language (such as OPS5) as a tool for encoding knowledge verbalizations concurrent with task performances, treating the former as an empirical knowledge data base (KDB) of heeded information. The (unstructured) production system that initially results from encoding is subjected to an analysis whose goals are determined by principles of program synthesis such as: locus of knowledge, locus of control, rule disabling/enabling patterns.

Resynthesis serves the purpose of compacting the knowledge encoding achieved in the (initial) production system representation by using separate goal and object memories and, possibly, hybrid inferencing (freely mixing forward and backward chaining). Potentially, re-synthesis can serve as the basis of translating optimized production systems into any programming language whatsoever.

It is shown that the bottleneck of this *decompilation approach* to knowledge acquisition lies in the present lack of means for the automatic derivation of encoding vocabularies (from empirical knowledge data), and the lack of means for automatic, or semi-automatic, utterance encoding.

It is shown generally that the task of modeling expert knowledge for building knowledge-based systems can be automated to the extent that it proves possible to develop intelligent tools for dealing with utterance encoding and production system analysis, as well as production system (re-) synthesis,-- a problem of automatic programming.

In short, it is demonstrated that problems in applied artificial intelligence presuppose novel solutions to theoretical artificial intelligence problems, especially in natural language understanding, automatic programming, and inductive theory (model) formation.

# TABLE OF CONTENTS

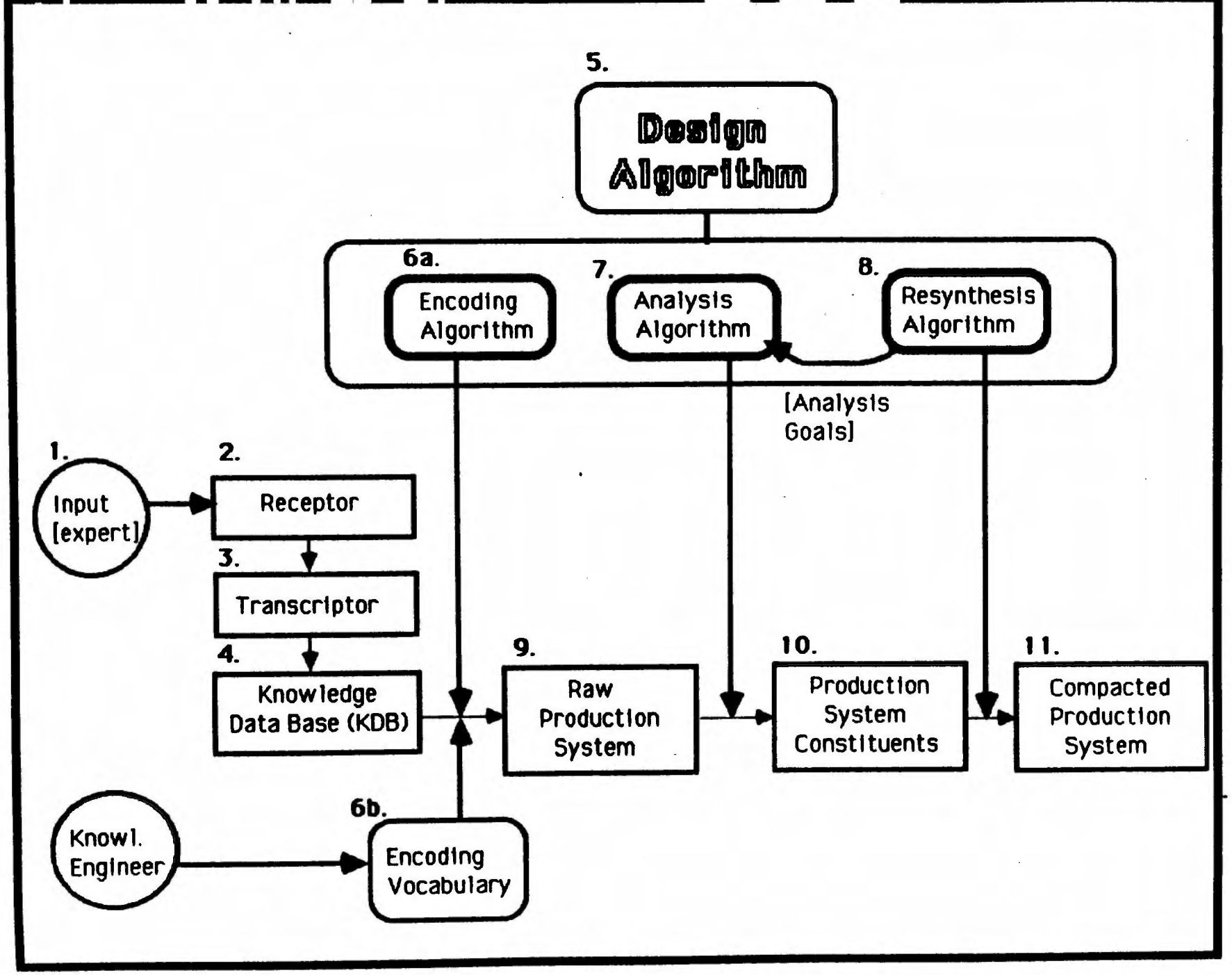
I. INTRODUCTION

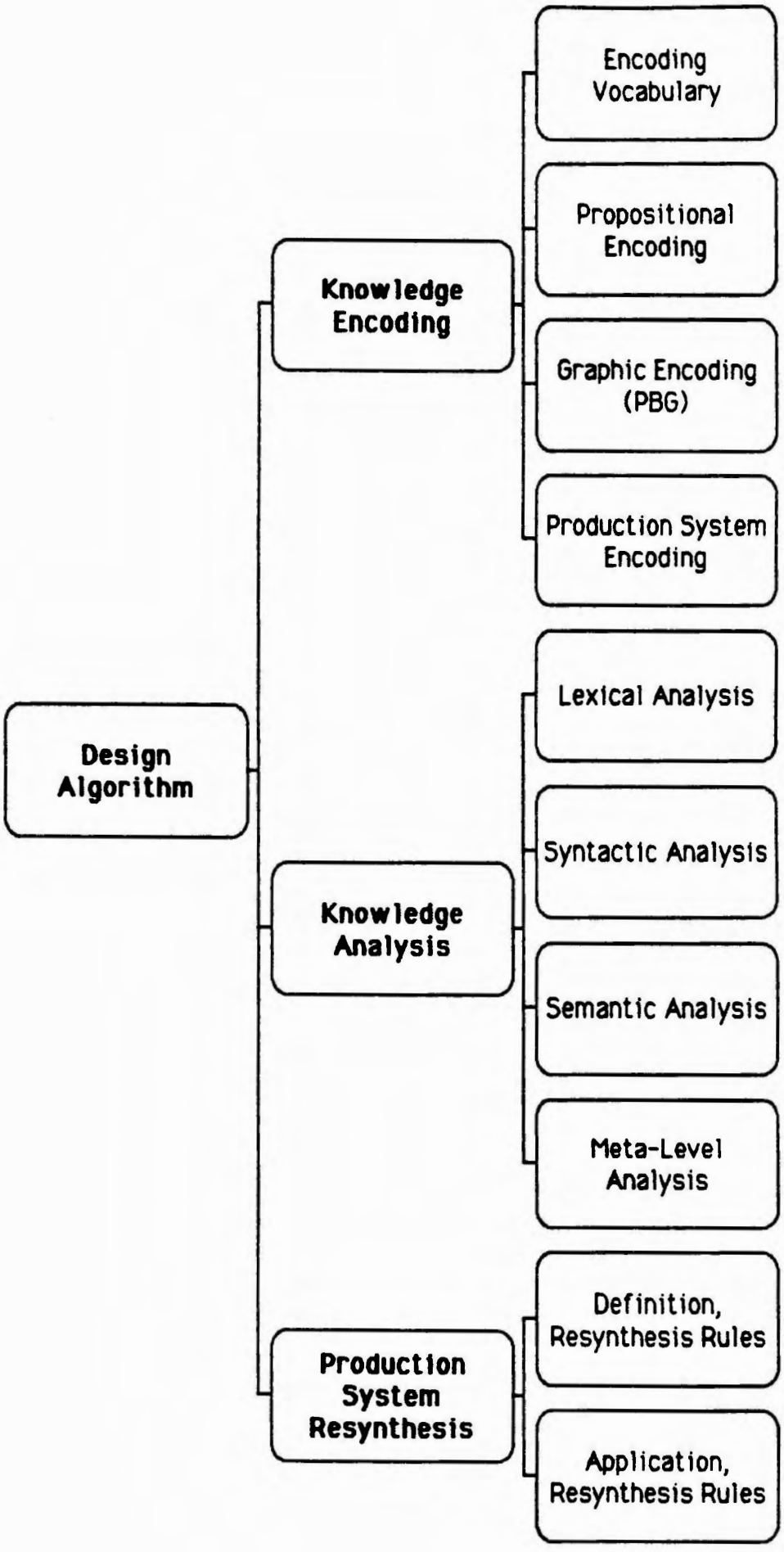
II. KNOWLEDGE ENCODING

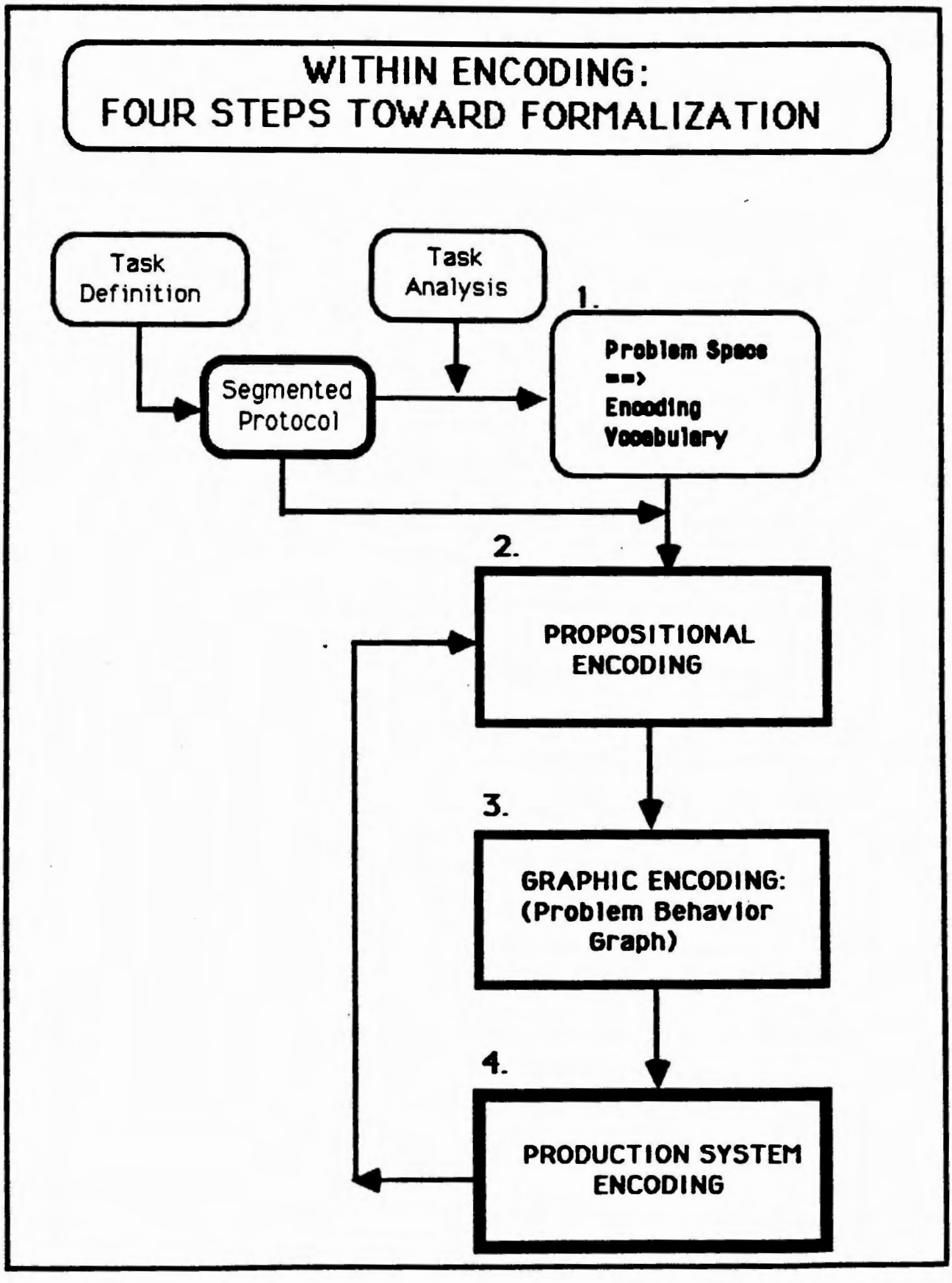
III. PRODUCTION SYSTEM ANALYSIS

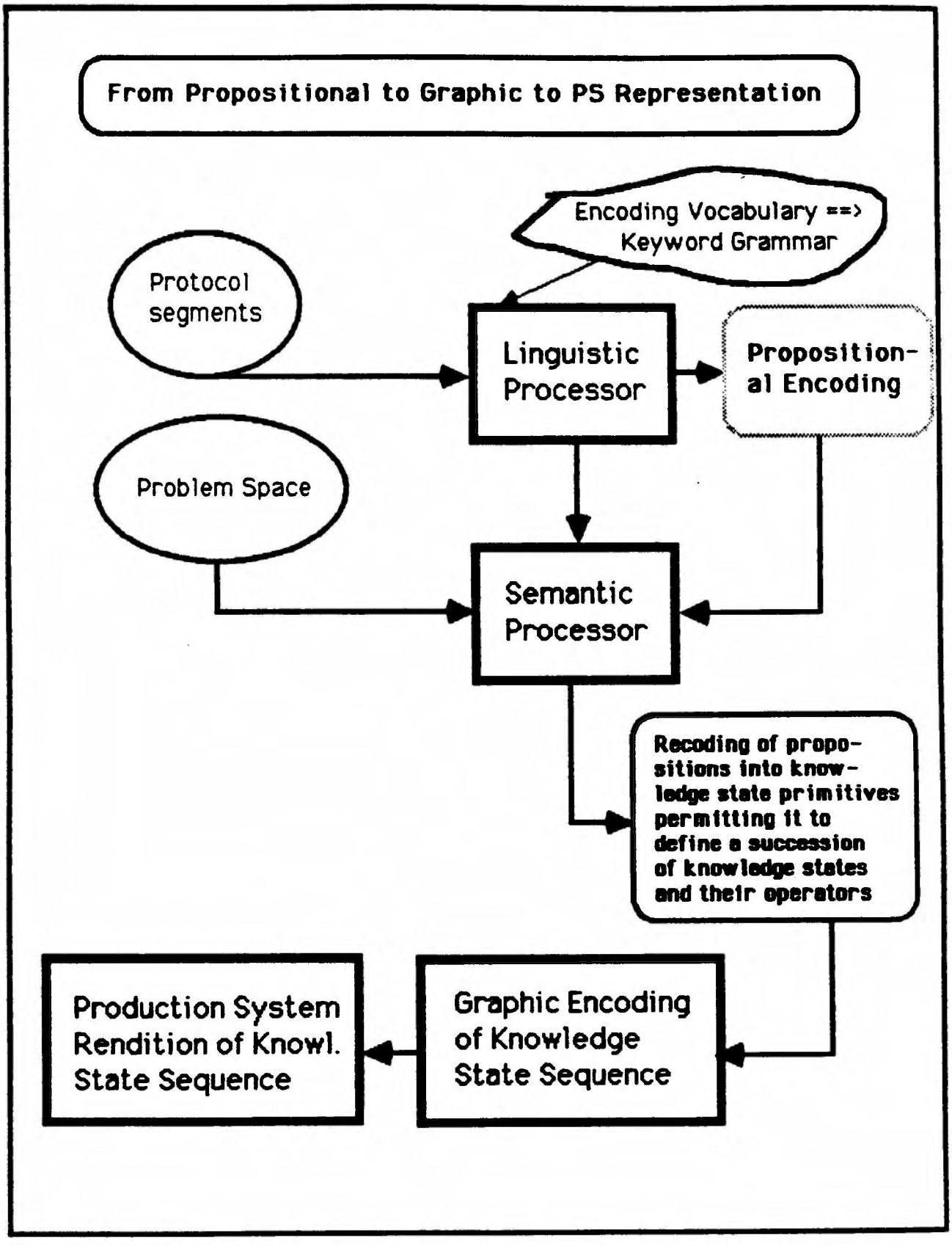
IV. PRODUCTION SYSTEM RE-SYNTHESIS

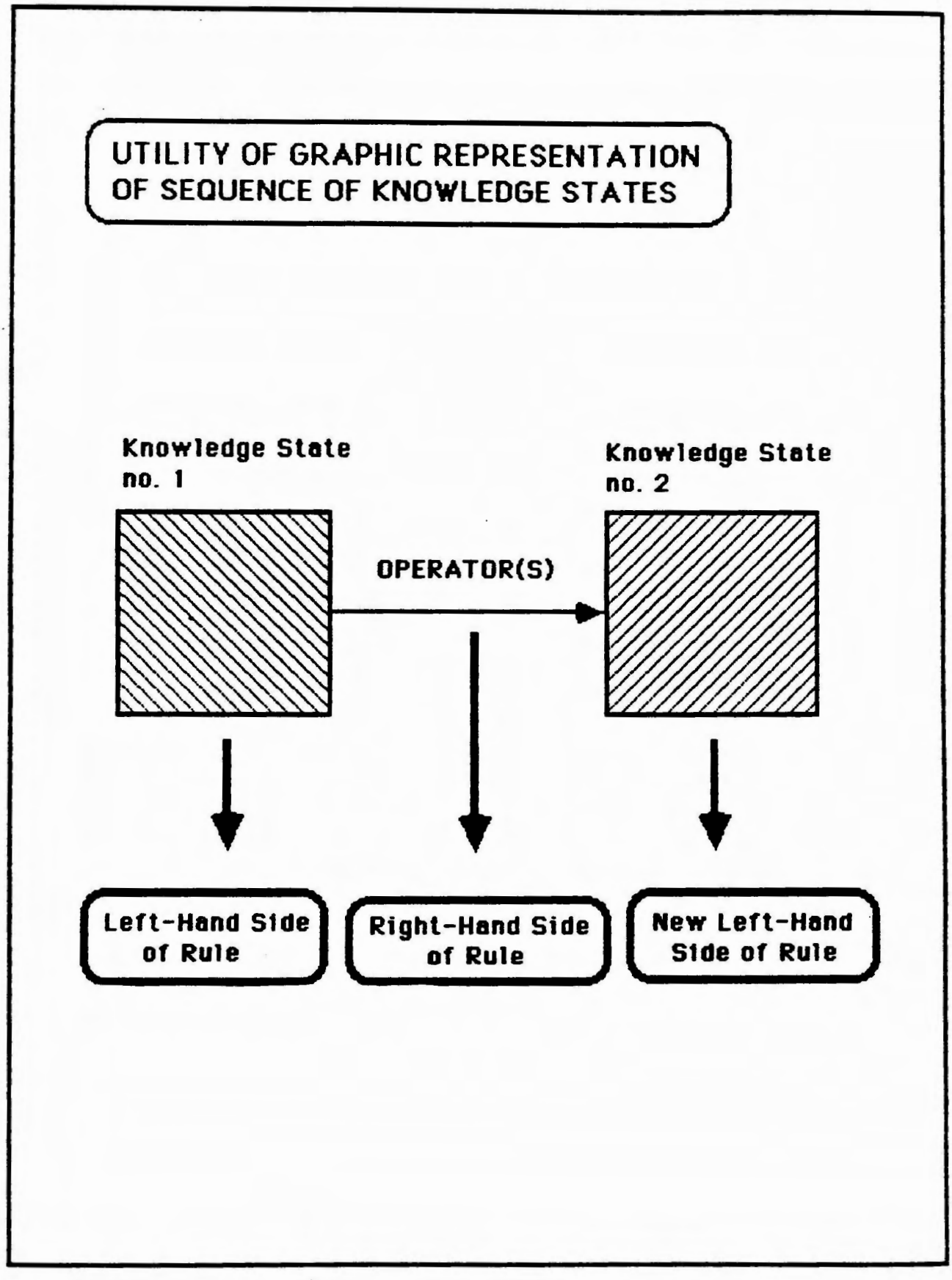
V. CONCLUSION

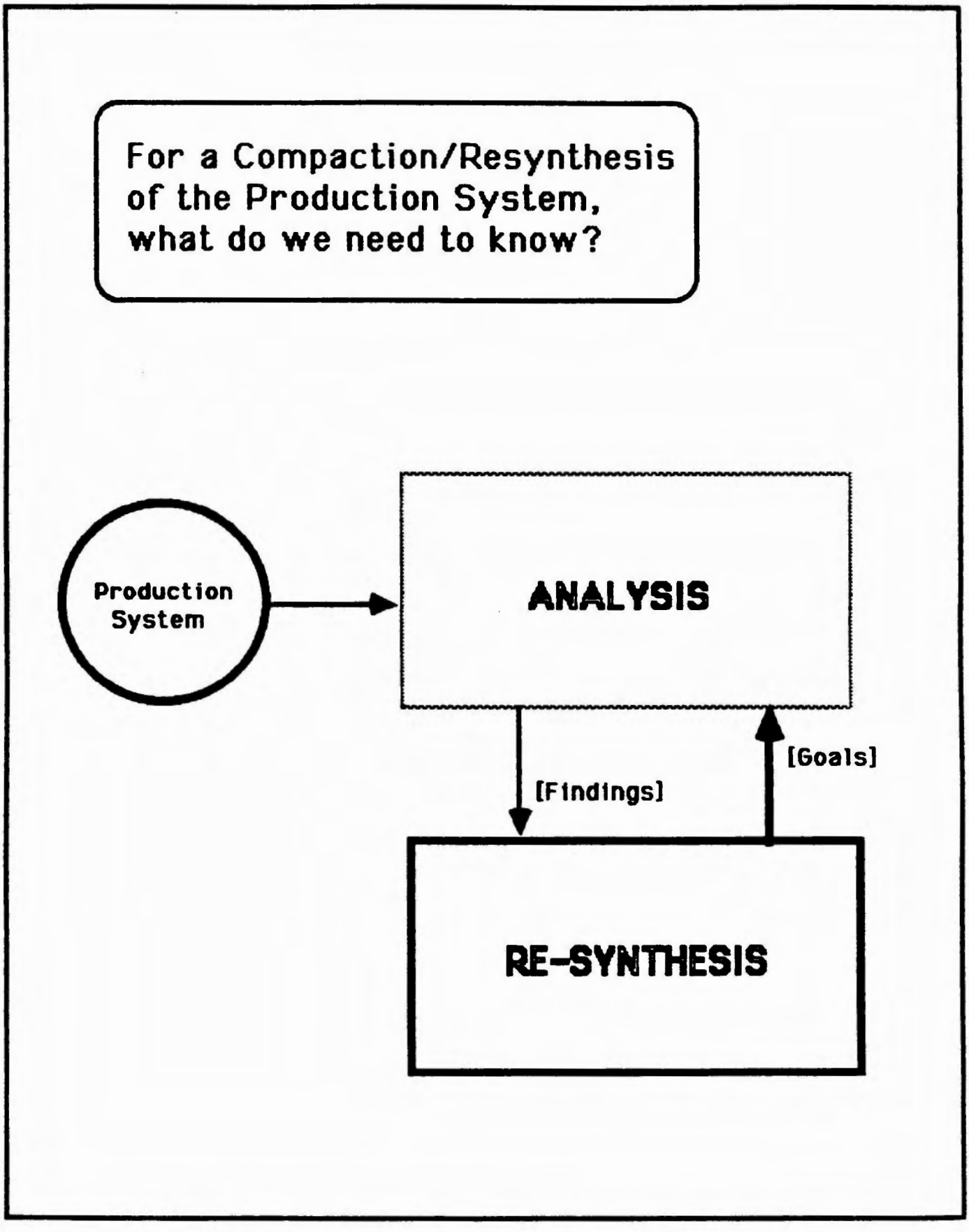

5.
Design Algorithm
6a.
Encoding Algorithm
7.
Analysis Algorithm
8.
Resynthesis Algorithm
[Analysis Goals]
1.
Input [expert]
2.
Receptor
3.
Transcriptor
4.
Knowledge Data Base (KDB)
9.
Raw Production System
10.
Production System Constituents
11.
Compacted Production System
Knowl. Engineer
6b.
Encoding Vocabulary

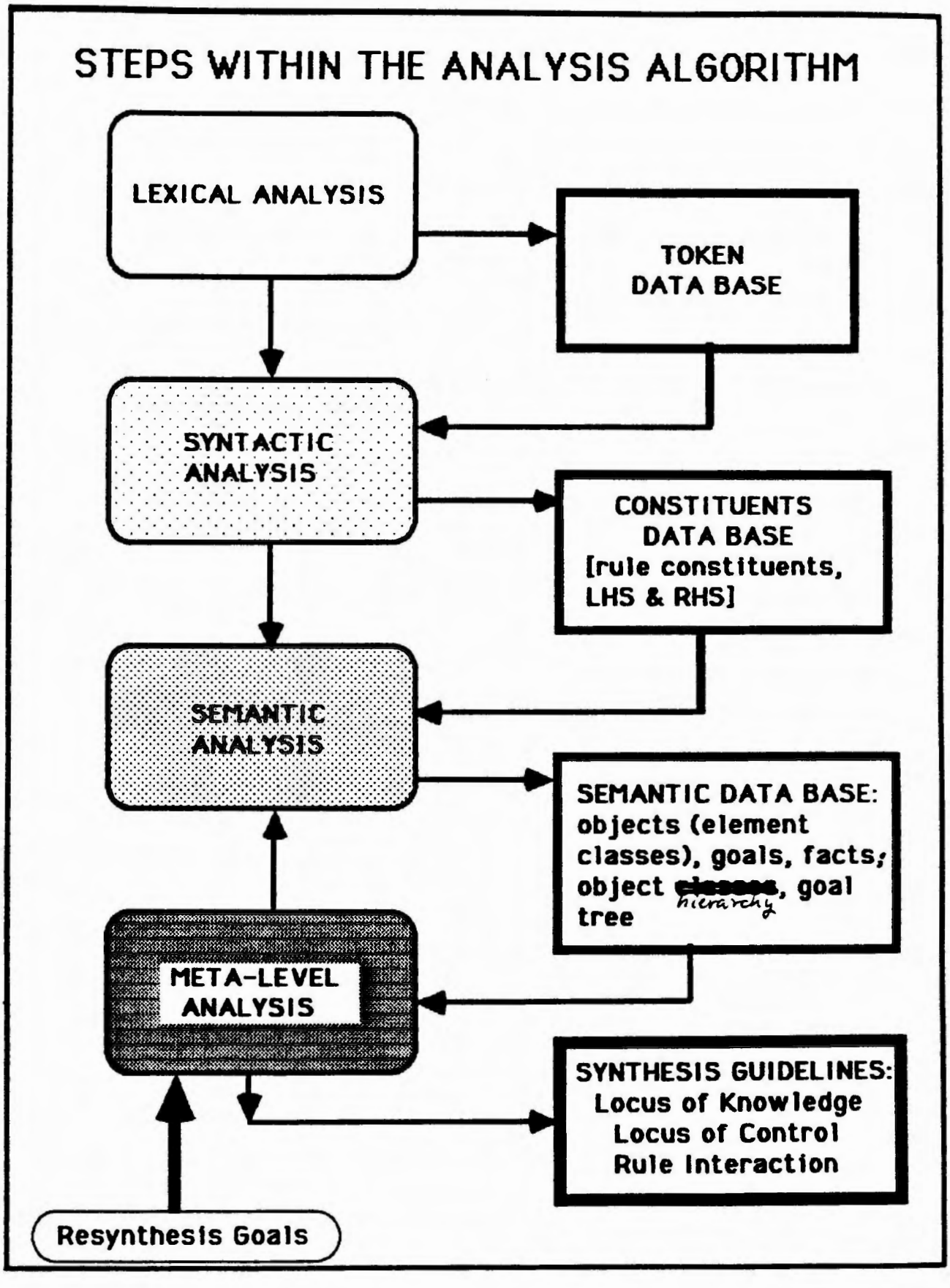

Design Algorithm
Knowledge Encoding
Encoding Vocabulary
Propositional Encoding
Graphic Encoding (PBG)
Production System Encoding
Knowledge Analysis
Lexical Analysis
Syntactic Analysis
Semantic Analysis
Meta-Level Analysis
Production System Resynthesis
Definition, Resynthesis Rules
Application, Resynthesis Rules

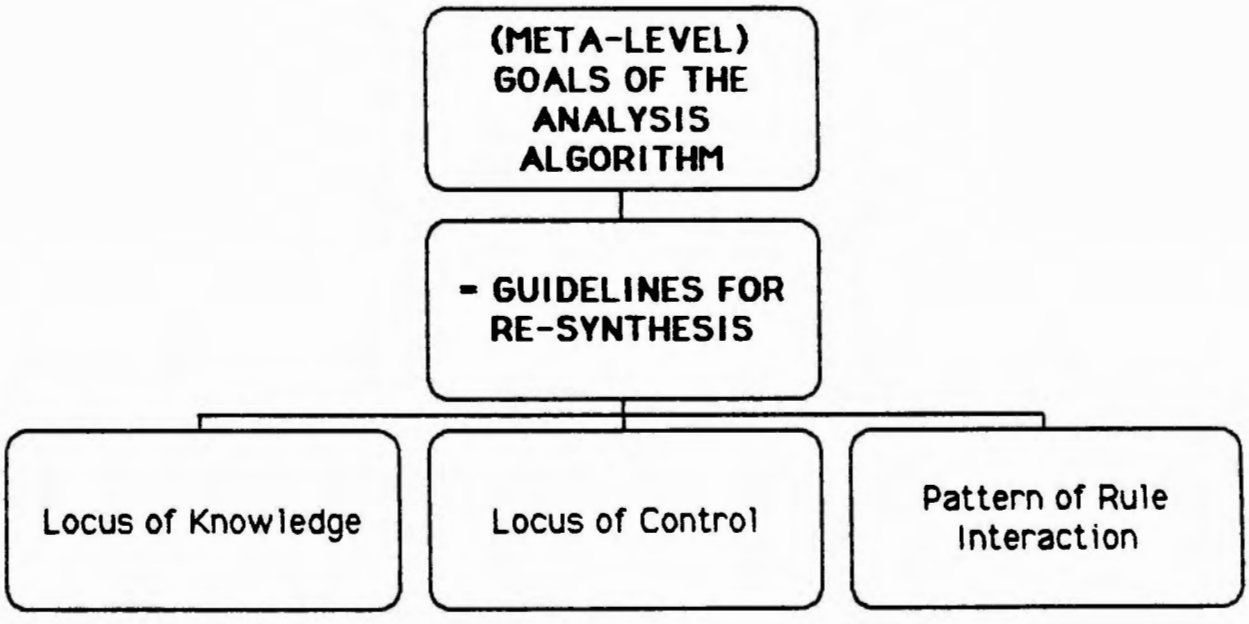

WITHIN ENCODING:
FOUR STEPS TOWARD FORMALIZATION
Task Definition
Task Analysis
1.
Problem Space ==> Encoding Vocabulary
Segmented Protocol
2.
PROPOSITIONAL ENCODING
3.
GRAPHIC ENCODING: (Problem Behavior Graph)
4.
PRODUCTION SYSTEM ENCODING


From Propositional to Graphic to PS Representation
Encoding Vocabulary ==>
Keyword Grammar
Protocol segments
Linguistic Processor
Proposition-al Encoding
Problem Space
Semantic Processor
Recoding of propo-sitions into know-ledge state primitives permitting it to define a succession of knowledge states and their operators
Production System Rendition of Knowl. State Sequence
Graphic Encoding of Knowledge State Sequence

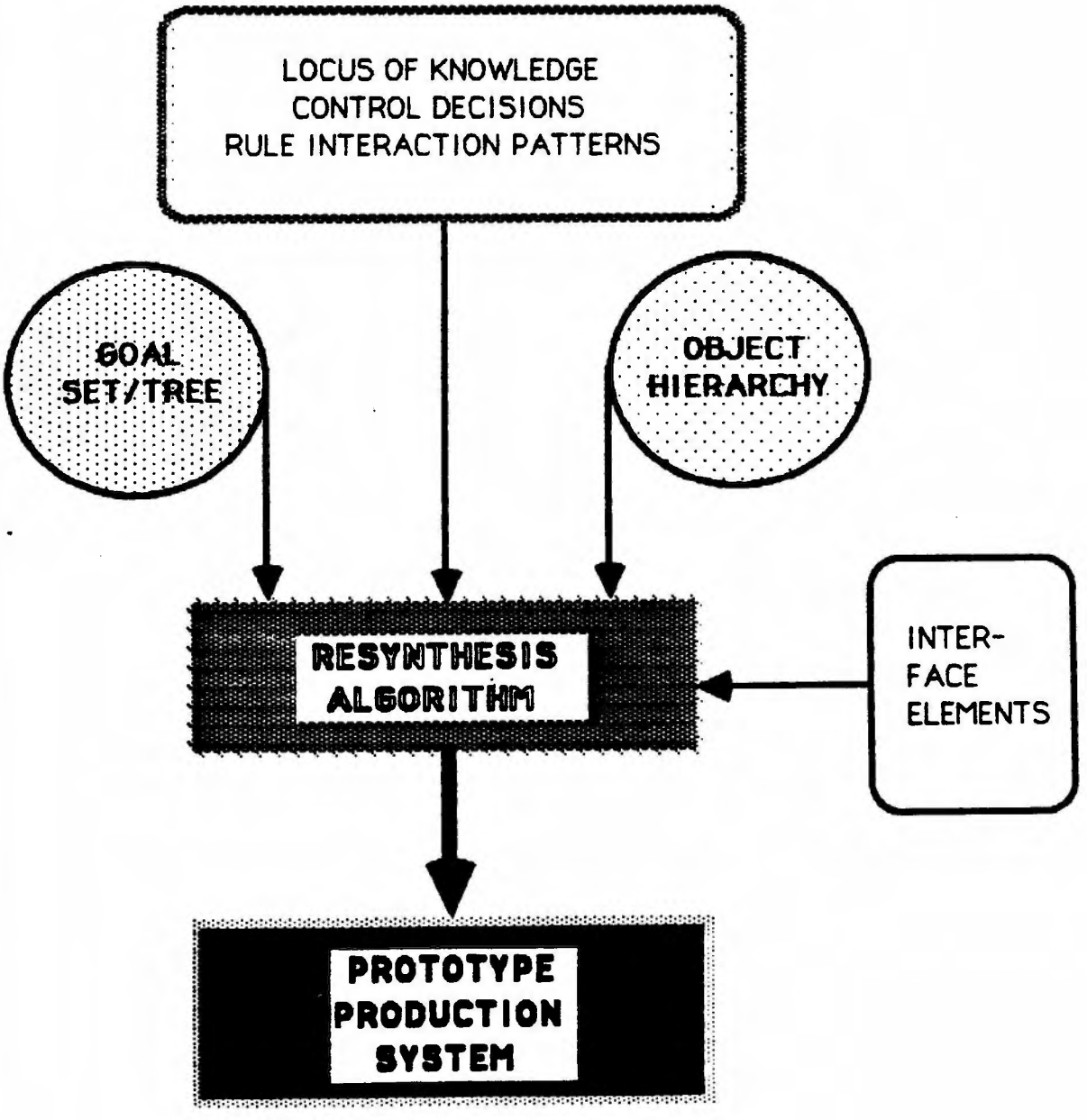

UTILITY OF GRAPHIC REPRESENTATION
OF SEQUENCE OF KNOWLEDGE STATES
Knowledge State
no. 1
Knowledge State
no. 2
OPERATOR(S)
Left-Hand Side
of Rule
Right-Hand Side
of Rule
New Left-Hand
Side of Rule


For a Compaction/Resynthesis of the Production System, what do we need to know?
Production System
ANALYSIS
[Findings]
[Goals]
RE-SYNTHESIS


STEPS WITHIN THE ANALYSIS ALGORITHM
LEXICAL ANALYSIS
TOKEN DATA BASE
SYNTACTIC ANALYSIS
CONSTITUENTS DATA BASE [rule constituents, LHS & RHS]
SEMANTIC ANALYSIS
SEMANTIC DATA BASE: objects (element classes), goals, facts; object hierarchy, goal tree
META-LEVEL ANALYSIS
SYNTHESIS GUIDELINES: Locus of Knowledge Locus of Control Rule Interaction
Resynthesis Goals


(META-LEVEL) GOALS OF THE ANALYSIS ALGORITHM
- GUIDELINES FOR RE-SYNTHESIS
Locus of Knowledge
Locus of Control
Pattern of Rule Interaction

# INPUTS TO AND OUTPUTS FROM THE RESYNTHESIS ALGORITHM



Problem Summary

## PROBLEMS AWAITING SOLUTION

1. TOOLS FOR THE AUTOMATIC DERIVATION OF ENCODING VOCABULARIES (KEYWORD GRAMMARS) FROM EMPIRICAL KNOWLEDGE DATA
   ==> a problem of natural language understanding

2. TOOLS FOR AUTOMATIC, OR SEMI-AUTOMATIC, UTTERANCE ENCODING
   ==> a problem of automated natural language formalization

3. TOOLS FOR AUTOMATIC PRODUCTION SYSTEM ANALYSIS DEPENDENT UPON VARIABLE (RE-SYNTHESIS) GOALS
   ==> a problem of "intelligent" program code analysis

4. TOOLS FOR PRODUCTION SYSTEM (RE-) SYNTHESIS
   ==> a problem in automatic programming

## BIBLIOGRAPHY

Brownston, L. et al. (1985). PROGRAMMING EXPERT SYSTEMS IN OPS5: An Introduction to Rule-Based Programming. Reading, Massachusetts: Addison-Wesley.

Brachman, R.J., H.J. Levesque (1985). READINGS IN KNOWLEDGE REPRESENTATION. Los Altos, California: Morgan Kaufmann Publishers, Inc.

Ericsson, K.A., Simon, H.A. (1983). PROTOCOL ANALYSIS: VERBAL REPORTS AS DATA. Cambridge, Massachusetts: The M.I.T. Press.

Harmon, P., D. King (1985). EXPERT SYSTEMS IN BUSINESS. New York City, New York: John Wiley & Sons.

Hart, A. (1986). KNOWLEDGE ACQUISITION FOR EXPERT SYSTEMS. New York City, N.Y.: MacGraw-Hill.

Jackson, P. (1986). INTRODUCTION TO EXPERT SYSTEMS. Reading, Massachusetts: Addison-Wesley.

Laske, O.E. (1986). "On Competence and Performance Notions in Expert System Design: A Critique of Rapid Prototyping." *Proceedings*, Sixth International Workshop on Expert Systems and their Applications. Avignon, France: Agence de l'Informatique. (J.-C. Rault, editor), 1:257-297.

Laske, O.E. (1987a). "A Three-Phase Approach to Building Knowledge-Based Systems". Unpublished manuscript.

Laske, O.E. (1987b). "A Course in Applied Epistemology." Technical Report, Center for Applied A.I., Arthur D. Little, Inc., Cambridge, Massachusetts, U.S.A.

Partridge, D. (1986). ARTIFICIAL INTELLIGENCE: Applications in the future of software engineering. Chichester, England: Ellis Horwood Ltd.

Winograd, T., F. Flores (1986). UNDERSTANDING COMPUTERS AND COGNITION: A New Foundation for Design. Norwood, N.J.: Ablex Publishing Corporation.

# The Role of Ontology in Concept Acquisition for Knowledge-Based Systems

Ira Monarch
Department of Computer Science
Sergei Nirenburg
Machine Translation Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
12 August 1987

## Abstract

We argue that a resident world model or ontology is indispensable for any general-purpose knowledge acquisition methodology used in developing expert systems, knowledge-based information retrieval, and natural language processing. We critically examine the various views on the nature of knowledge in such systems, starting with the two primary approaches to knowledge bases - rule bases and frame bases. We argue that frame systems are necessary for adequately representing world and domain models. We then discuss the major interpretations of frame systems regarding their logical form or lack of it and their fundamental cognitive or semantic units. Emphasizing cognitive and linguistic factors, we then propose a knowledge level methodology for building domain models based on an underlying ontology. Finally, we describe ONTOS, a knowledge acquisition system we developed which aids in building domain models on the basis of our ontology. The first large-scale application for this system is acquiring the concept lexicon for a knowledge-based machine translation project at CMU.

## Table of Contents

1. Knowledge Acquisition Framed 1
1.1. The Role of Ontology in Knowledge Acquisition 1
1.2. ONTOS: An Ontologically Based Knowledge Acquisition System 2
2. Using ONTOS 2
2.1. An Example of ONTOS in Use 2
2.2. Interaction at the Interface 3
2.3. ONTOS Initiates Interaction 3
3. Functionality 3
4. Advantages of ONTOS 4
4.1. Other Views of Ontology and Semantics 5
4.2. Use in Various Applications 5
4.3. Comparison with Other Systems 6

## 1. Knowledge Acquisition Framed

Many AI research projects at CMU and elsewhere need to acquire and maintain variably sized knowledge bases. More and more of these projects are using frame representation languages. Although many knowledge acquisition systems, especially those for building and maintaining expert systems, are used for acquiring rule-based knowledge, frame representation languages are now used to supplement these rule bases. Frame representations are used because they are easier to debug and modify and because they can be used to structure knowledge according to high level organizational principles [Fikes & Kehler 85]. Frame representations conceptually interpret the terminology used in rules enabling rule-based systems to "know what their rules are about." [Lenat, Prakash, & Shepherd 85] Many groups at CMU are using a family of frame representation languages for systems which:

- Analyze and generate natural language for machine translation - Center for Machine Translation [Carbonell & Tomita 87]
- Aid Information Retrieval - MedSORT [Carbonell, Evans, Scott, and Thomason 85] and CoalSORT [Monarch & Carbonell 87]
- Reason and Problem Solve - FERMI [Larkin, Reif, and Carbonell 86]
- Plan and Learn - PRODIGY [Carbonell & Gil 87]

This situation has made it attractive, perhaps even necessary, to develop a general frame-based knowledge acquisition system for such projects.

### 1.1. The Role of Ontology in Knowledge Acquisition

Frame representation languages need to be interpreted [Brachman 79]. Frame systems which are useful for acquiring knowledge, as well as representing it, need to go one step further. They need a resident ontology over and above a knowledge representation language as the basis for constructing domain models, whatever the domain.

Methodologically significant problems arise with respect to the universality of ontology; different ontological judgments and decisions are typically made by various researchers. This might lead some to confuse ontology with epistemology. It is therefore important to distinguish the question of **what** is it you know when you know it from the question of **whether** you know something when you believe it. One does not have to wait until all the truths are in to avoid ontological promiscuity.[1] A number of research groups are beginning to gain experience building ontologies.[2] A viable ontologically based knowledge acquisition system does not need to rely on any one of these accounts. In fact we make no restrictive claim for the ontology we present below as being the final word. Our system supports changing the given ontology and propagating these changes down the hierarchy. A viable ontologically based knowledge acquisition system should be hypertextual and provide ontological alternatives depending on the domain. However, we do believe that the ontological analyses being undertaken will converge significantly and provide a commonly agreed upon ontological basis for concept acquisition.

---

[1] As Jerry Hobbs implies in his paper "Ontological Promiscuity" [Hobbs 85]

[2] These include those working in qualitative physics such as DeKleer [DeKleer & Brown 85], Forbus [Forbus 85], and Hayes [Hayes, 85]; those recommending ontological analysis for knowledge level engineering - Freiling [Alexander, Freiling, et al 86]; and those proposing a computational semantics for natural language - Nirenburg and Raskin [Nirenburg & Raskin 87], Barwise and Perry [Barwise & Perry 83], Creary and Pollard [Creary & Pollard 85], and Hobbs [Hobbs, et al 86]. This is only a representative sample.

### 1.2. ONTOS: An Ontologically Based Knowledge Acquisition System

Our system, ONTOS, is an intelligent interactive knowledge acquisition tool that supports the acquisition of domain models on the basis of a pre-defined ontology. It will be used initially to construct on-line dictionaries and thesauri -- to structure, index, and represent semantic information for use in natural language processing tasks. We also intend to make it the basis for either a knowledge-based information retrieval or hypertext system. It is also intended to be an integral part of a system to study cognitive agency, planning, and eventually concept learning. In what follows we outline the:

- types of knowledge it uses and acquires
- design of the system
- character of the interface
- major differences from other knowledge acquisition systems.

ONTOS is used for knowledge entering, not knowledge extraction. We assume that candidate concepts have already been introduced (like an index to the back of a book). We believe that there are many interesting and significant problems to solve, though this is often overlooked, even after knowledge has been extracted from an expert or a text. This is not to say that we ignore the knowledge extraction problem. We are planning a major research thrust into the problem of indexes into episodic memory for texts. An important part of this effort is to enhance ONTOS so that it will aid knowledge extraction from texts.

## 2. Using ONTOS

Making ontological categories an intrinsic part of a knowledge acquisition system enables users to construct domain concepts on the basis of a system of analyzed ontological categories already given. This analyzed system of categories provides exemplars which users can copy and modify as they build a conceptual model of their domain. Users do not have to start from scratch. Of course these categories must be presented as part of an intelligent interactive system. This requires:

- appropriate hardware - state of the art high-powered workstations with appropriate software environment and adequate memory
- front end flexibility of interface including such features as natural language interaction, graphic presentation and browsing, frame editing and browsing
- backend consistency checks

### 2.1. An Example of ONTOS In Use

ONTOS is still in the beginning stages of development. At this point only the very top levels of the ontology are taking shape. The following are the present top level categories:

- Objects
- Relations and Properties
- States of Affairs
- Time
- Space

Fragments of the first three are diagrammed in figures 4-1, 4-2, and 4-3 pages 8, 9, and 10). These figures are pictures of ONTOS screens. The screens are part of an interface for viewing and building conceptual networks. During development concept acquisition will proceed bottom-up as well as top-down. Various prototype domains will be explored for indications of the kind of ontological support

needed.

### 2.2. Interaction at the Interface

A session with ONTOS begins with two interactive windows - the Graph Browser: Top of Hierarchy and a File Operations menu. A mouse is used to point to nodes in the Graph Browser and initiate different events:

- Nodes can be selected for expansion one level down from the selected node.
- A Frame editor can be opened with a selected frame to be edited.

A Standing menu for file system interaction, including printing the contents of a window are shown in the upper right of figure 4-1 page 8. Icons for system windows are also shown at the bottom left. Graphic editing to introduce new nodes into is-a and part-of hierarchies is planned. The first example screen shows an interactive state in which the concept **artifact** is being entered into the knowledge base. The Partial Object Hierarchy shows that **artifact**, upon insertion, is assigned a place in the object hierarchy.[3] The Frame Browser/Editor in the upper center shows the internal slot-value structure of the **artifact** node. The majority of these slots have been inherited from nodes above **artifact**. A new slot-value, *made-by:* **human**, has been added.[4] Neither the slot-name nor the value have been defined as concepts. This causes ONTOS to initiate further interaction.

### 2.3. ONTOS Initiates Interaction

ONTOS initiates interaction with a knowledge enterer in the ONTOS Command/Prompt area. This area is shown at the bottom right half of the screen in each of the figures. In figure 4-2 page 9, ONTOS is alerting the enterer that **made-by** and **human** have not yet been defined. It enables the enterer to add new concepts which cover the new slot and value entered with the **artifact** frame. ONTOS also can link the new concepts to a node in the **is-a** hierarchy specified by the enterer. If linked, ONTOS redraws the hierarchy in any windows affected by the new linkage, and opens a Frame Editing window with the new concept and the option to inherit information from above it in the hierarchy.[5] Figure 4-3 page 10 shows the enterer adding the value **plan** to the intention slot of **human**, inserting the frame, and initiating the specification of **plan** as a concept.

We will be adding natural language interaction capabilities to the system at the Command/Prompt line. We will be experimenting with learning by being told, as was done in the NANOKLAUS system described by [Haas & Hendrix 83].

## 3. Functionality

We are in the process of developing programs that will compare an edited node with its relations, i.e.,

- Parent(s)
- Children
- Siblings

and alert the knowledge enterer if it is not distinct in some way. Other sorts of checks and prompts are

---

[3]These hierarchies will no doubt become tangled as work proceeds.

[4]We will soon be distinguishing inherited information from locally specified information by highlighting the latter. The system also allows the names of some slots and values to be changed as the need arises.

[5]Even if no link is specified, ONTOS can still open a Frame Editing window, though there will be no option to inherit information.

being developed as well. For example:

1. If a newly created node has siblings, the system will check whether any of these siblings are subsumable under it or whether it is subsumable under them. If so, the system will prompt the user with suggestions. See discussion in item 6 below on completing incomplete entries automatically.
2. When a user adds a new slot and value, the system will check if they are specified in the concept hierarchy or or in the latter case if it is the value of a property. If not, the system will prompt a knowledge enterer:
    a. to add the slot either as a relation or as a property
    b. to add the value either as a concept or as a range of a property.
3. The system will check the appropriateness of a newly ascribed value for a given slot, either with respect to the meaning of the slot or with respect to inherited fillers of the slot. The following are only initial examples. The system will prompt the user:
    a. if a concept is incorrectly specified to fill a property slot or a property value is incorrectly specified to fill a relation slot - Don't Allow
    b. if a local slot value is specified which overrides an inheritable slot value; no need to prompt if, the local slot value IS part of the range or disjuncts of the parent slot value - Warning Only.
4. The system will create inverses automatically through interaction with a user. It will:
    a. Prompt the user to specify the name of the inverse when adding a new relation
    b. In certain situations the system will automatically add inverse relations, provided that they are already defined (see above).
5. The system will automatically transform spatial units specified by a user into standard numerical units. This will include recognizing different quantitative units and their common abbreviations. The aim here is to promote the standardization of property values. Standard numerical property values for some other common quantitative properties are being developed and explored.
6. The system will complete an incomplete entry automatically. This is an expansion of item 1 and a longer term goal.

    If a user enters a node without an is-a link, the system will determine if there are any nodes in the hierarchy which would be good candidates to be thus linked and prompt the user. For example, if a given node has significant slots which intersect with the new node and all of the slot values of the latter are subranges of the former, the former might be a good candidate for the latter to be is-a. The story is more complicated than this, but this is the general idea. Other candidate slots and values may also be determinable.
7. We are discussing the problems involved when several or more people work together to build domain models.

## 4. Advantages of ONTOS

We will now describe some of the advantages of ONTOS, especially with respect to other views of ontology and semantics - both competing and similar. We will also explore its use in various applications and compare it to systems performing some of the same functions. The following is only meant as an introduction to some relevant research and our views on it.

### 4.1. Other Views of Ontology and Semantics

- Logic and Ontology - Frames enable automatic subsumption of structured types seemingly without many ontological scruples [Levesque & Brachman 85]. But knowledge acquisition is more than just automatic subsumption. We believe that for automatic subsumption to procede properly, ontological categories, including what are called natural kinds, have to be specified first. This includes basic slots or roles, that is basic properties and relations. Our strategy is to provide a lot of top level support, so that adding new concepts becomes a focused and simple editing task through the use of inheritance. Once a large portion of ontology and a given domain is specified automatic subsumption becomes more useful. Hayes [Hayes, 85] and Hobbs [Hobbs, et al 86] have argued for axiomatizing commonsense physics and metaphysics respectively. We agree with much of the thrust of their work. However, their emphasis on axiomatization has had them de-emphasizing, or at least not acknowledging, the problems of large scale knowledge acquisition and the sort of frame-based approaches we recommend for dealing with these problems. We think this is behind Hobbs somewhat cavalier attitude toward ontology as expressed in his paper "Ontological Promiscuity"
- Cognitive Models - What Are the Cognitive Units? - Can diversified knowledge structures such as scripts, plans, themes be specified in a uniform way? [Galambos, Abelson, Black 86]. Our view is that they can on the basis of ontological categories.
- Linguistic Semantics - What is the role of ontology in frame semantics? [Nirenburg & Raskin 87], [Fillmore 82], and others) Our view is that lexical semantics is dependent on ontology. This view is not very different from Hobbes' view in [Hobbs, et al 86]: "By developing core theories of several fundamental phenomena and defining lexical items within these theories ... we are able to cope with complexities of word meaning that have hitherto escaped lexical semanticists, within a framework that gives full scope to the planning and reasoning processes that manipulate representations of word meaning." As has been pointed out, we differ in emphasizing a frame ontology with little emphasis on axiomatization and theorem proving. Our main goal is to integrate what Hobbs calls "core theories" in an ontology and knowledge acquisition system which facilitates the building of large-scale knowledge bases.

### 4.2. Use in Various Applications

- Natural Language Processing - Acquiring lexicons for analysis and generation; employing machine readable dictionaries to assist in such tasks, (see citations above, especially Nire87). There is a passive on-line dictionary and encyclopedia already available at CMU.
- Information Retrieval - Researcher [Lebowitz 85], Rabbit [Tou, Williams, Fikes, Henderson, and Malone 82], CoalSORT [Monarch & Carbonell 87], and MedSORT [Carbonell, Evans, Scott, and Thomason 85]
- Reasoning and Problem Solving in Complex Domains - CYC [Lenat, Prakash, & Shepherd 85], Prompt [Addaanki & Davis 85], Fermi [Larkin, Reif, and Carbonell 86]
- Learning - Genesis [Mooney and DeJong 85], Ferret [Mauldin 85], Prodigy [Carbonell & Gil 87]

One of the authors[6] has been exploring the idea of using frame-based ontologically interpreted semantic networks in hypertext systems. The idea is based on the analogy between semantic networks and hypertexts. If the chunks of text which are the nodes of a hypertext system are represented by iconic labeled nodes and the links by variously patterned and labeled lines, then the graphic network representing a hypertext is much like a semantic network.[7] In fact, the semantic network representation

---

[6] [Monarch 87]

[7] Some who have pointed out this analogy are [Corda & Facchetti 86] and [Conklin 87].

of the content of a printed text can be used as a hyper-index of that text by indexing the text into concept nodes of the network. This can be the basis for a hypertext system.[8] In such a system portions of text can be linked to other portions through being indexed by nodes in the semantic network. All texts represented by the same concept (e.g. synonyms) will be indexed by that concept. Frame representations of these nodes provide conceptual glosses of the portions of text they index. The links of the semantic network can also be used as the links connecting portions of text to one another. Interpretive glosses of these links can be based on the conceptualizations given them in the semantic network. Concepts which are, for example, **is-a** one another can represent corresponding texts which either **specify** or **generalize** the other. [Tucker, Nirenburg, & Raskin 86] Also, concepts such as actions and the corresponding activities they are **part-of** can represent corresponding texts which are **analyzed-parts** or **synthesized-wholes** of the other(s).

Different texts will have different graphic networks, more or less detailed, representing them. Even if two texts share one or more concepts, their graphic networks will be different. These graphic networks provide a basis for classifying and differentiating texts for the purposes of retrieval. They are also a basis for presenting texts in quick graphs so that they can be easily discriminated through browsing and selected for further investigation.

There are a number of research issues here. Many of them involve, broadly speaking, the relations between semantic and episodic memory. Building such a knowledge-based hypertext system is not merely a matter of applying completed research. We are aiming to build a system which will have practical uses, but which will be continually improved as research continues. We believe that such a knowledge-based hypertext system is an important tool for resolving cognitive problems of hypertext.

### 4.3. Comparison with Other Systems

1. Natural language interfaces to data bases - e.g., TELI [Ballard & Stumberger 86], TEAM [Grosz, Appelt, Martin, and Pereira 87], IRACQ [Ayuso, Shaked, & Weischedel 87]. We believe that a language independent concept lexicon management system is needed for lexical semantics and also for building on-line dictionaries useful for parsing in information retrieval. The above systems are too dependent on language specific analysis and have developed no principled strategy for language independent concept acquisition nor seem to have any desire to do so.

2. Generic knowledge acquisition tools:
   - CYC [Lenat, Prakash, & Shepherd 85]: The knowledge acquisition methodology guiding this project is too empirical. Text analysis of randomly selected encyclopedia articles is not based on any prior domain analysis nor on any higher level ontological analysis. An attempt is being made to represent all of commonsense, instead of proceeding domain by domain. While we recommend an intensive ontological or top-down analysis, we realize that some ontological distinctions will also come about bottom-up, through intensive domain analysis. However both of these seem to be missing here.
   - NIKL [Kaczmarek, Bates, & Robins 86]: There is good support for classification based reasoning, but no support for structuring categories. There is hardly any acknowledgement that this is necessary, except in a footnote made to appear

[8] It needs to be pointed out here that in the frame-based semantic network which is the framework of data structures on which our knowledge acquisition system is built, nodes can correspond to concepts which represent complex situations like activities which are goal-directed sequences of actions in specified contexts.

somewhat digressive.[9] There is no other mention of the problem of determining the ontological structure of the knowledge base and consequently there is no discussion of how to deal with this problem.

A main concern of ours has been to avoid haphazard knowledge acquisition for specialized tasks with little or no concern for reusability. We think the tools for knowledge acquisition described above do not avoid these problems because they are too dependent on the semantics of a specific language and they do not incorporate any ontological or domain analysis in their knowledge acquisition strategies.

---

[9] "A primitive concept or relation corresponds to the notion of a 'natural kind,' i.e., a predication that can only be determined by an oracle. To NIKL this means that no concept may be placed beneath this one in the hierarchy unless the concept specification explicitly says to do so."

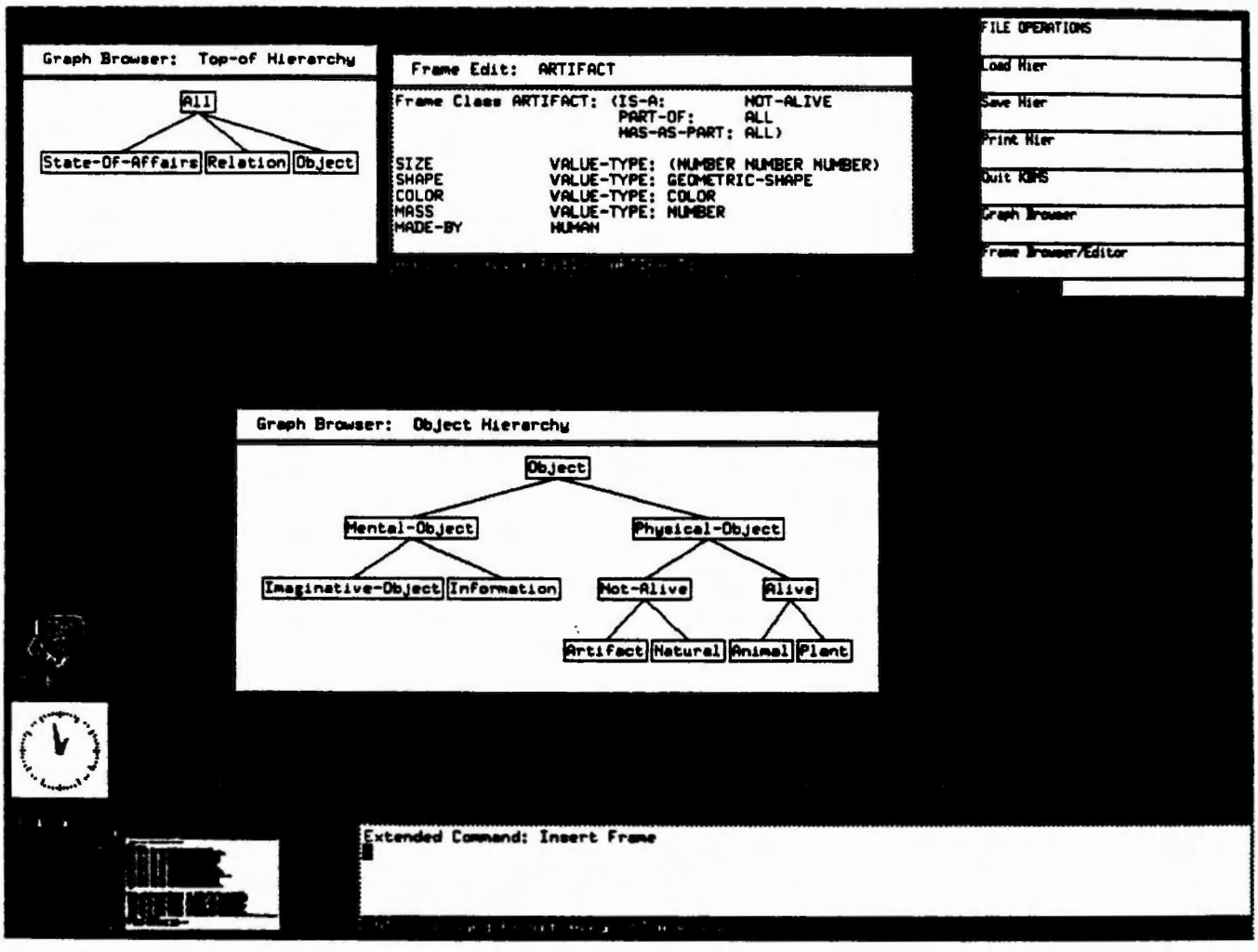



**Figure 4-1:** ONTOS Screen - Inserting ARTIFACT

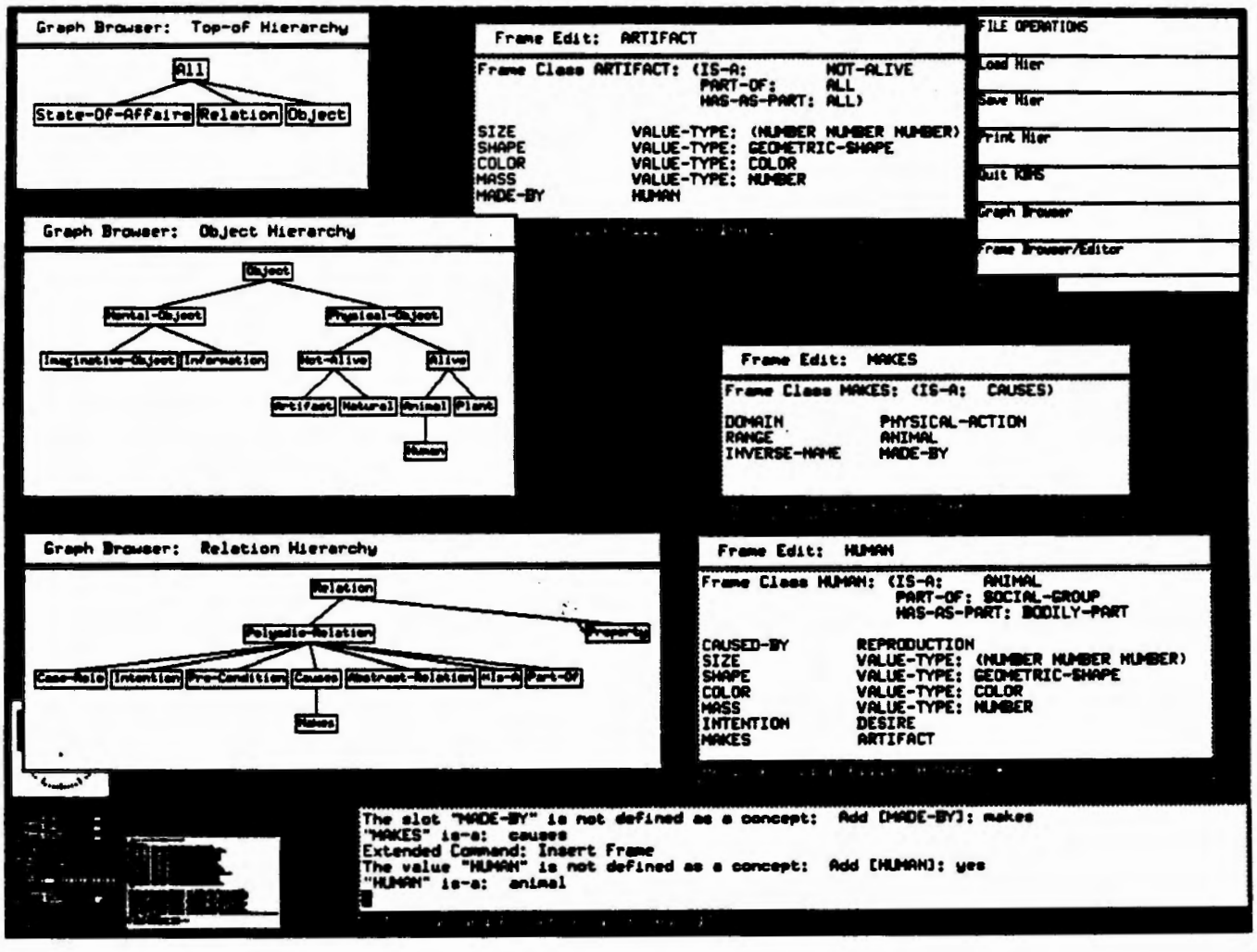



**Figure 4-2:** ONTOS Takes The Initiative - Adding MAKES and HUMAN

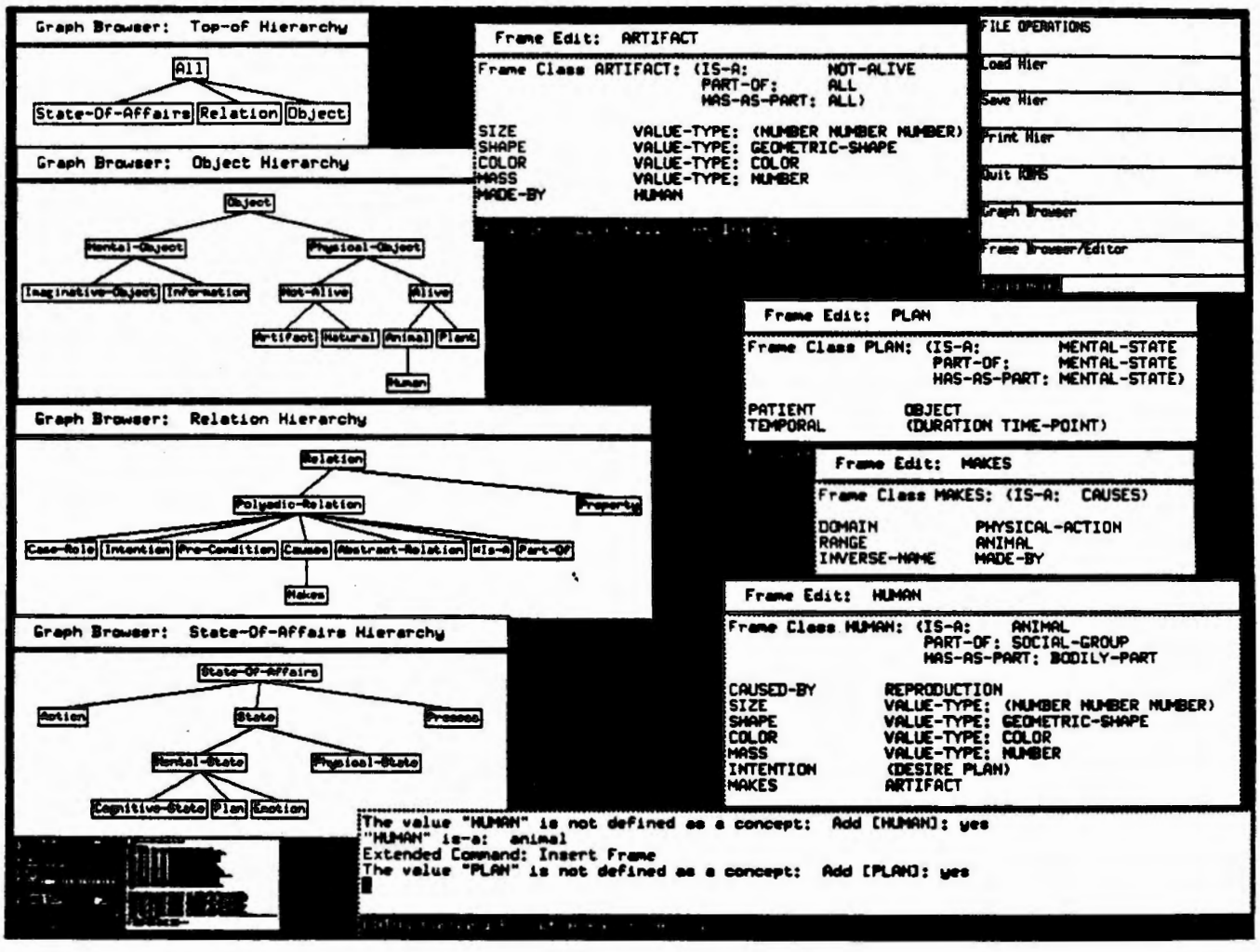



**Figure 4-3:** Showing More of the Hierarchy: Adding The Value PLAN

## References

[Addaanki & Davis 85]
Addanki, S., and Davis, E.
A Representation For Complex Physical Domains.
In *Proceedings IJCAI-85.* Los Angeles, CA, 1985.

[Alexander, Freiling, et al 86]
Alexander, J.H., Freiling, M.J., and et.al.
Knowledge Level Engineering: Ontological Analysis.
In *Proceedings AAAI-86.* Philadelphia, PA, 1986.

[Ayuso, Shaked, & Weischedel 87]
Ayuso, D.M., Shaked, V., and Weischedel, R.M.
An Environment for Acquiring Semantic Information.
In *Proceedings of the ACL-87 - to appear.* 1987.

[Ballard & Stumberger 86]
Ballard, B.W., and Stumberger, D.E.
Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor.
In *Proceedings of the ACL-86*, pages 20-29. 1986.

[Barwise & Perry 83]
Barwise, J., and Perry, J.
*Situations and Attitudes.*
MIT Press, 1983.

[Brachman 79] Brachman, R. J.
On the Epistemological Status of Semantic Networks.
In N. V. Findler (editor), *Associative Networks.* New York: Academic Press, 1979.

[Carbonell & Gil 87]
Carbonell, J.G., and Gil, Y.
*Learning by Experimentation.*
Technical Report, Carnegie-Mellon University, Computer Science Department, 1987.

[Carbonell & Tomita 87]
Carbonell,J.G. and Tomita, M.
Knowledge-Based Machine Translation: The CMU Approach.
In Nirenburg, S. (editor), *Machine Translation: Theoretical and Methodological Issues.* Cambridge University Press, 1987.

[Carbonell, Evans, Scott, and Thomason 85]
Carbonell, J.G., Evans, D.A., Scott, D.S., and Thomason, R.H.
*Final Report on the Automated Classification and Retrieval Project.*
Technical Report, CMU, December, 1985.

[Conklin 87] Conklin, J.
*A Survey of Hypertext.*
Technical Report STP-356-86, Rev. 1, Microelectronics and Computer Technology Corporation, February, 1987.

[Corda & Facchetti 86]
Corda, U. and Facchetti, G.
Concept Browser: A System for Interactive Creation of Dynamic Documentation.
In van Vliet, J.C. (editor), *Text Processing and Document Manipulation: Proceedings of the International Conference, University of Nottingham.* Cambridge University Press, 1986.

[Creary & Pollard 85]
Creary, L.G., and Pollard, C.J.
A Computational Semantics for Natural Language.
In *Proceedings of the ACL-85*, pages 172-179. 1985.

[DeKleer & Brown 85]
Dekleer, J. and Brown, J. S.
A Qualitative Physics Based on Confluences.
In Hobbs, J. R. and Moore, R. C. (editors), *Formal Theories of the Commonsense World*, pages 109-183. Ablex Pub. Co., Norwood, NJ, 1985.

[Fikes & Kehler 85]
Fikes, R., and Kehler, T.
The Role of Frame-based Representation in Reasoning.
*Communications of the ACM* 28(9):904-920, September, 1985.

[Fillmore 82]
Fillmore, C.J.
Frame Semantics.
In The Linguistic Society of Korea (editor), *Linguistics in the Morning Calm*. Hanshin, 1982.

[Forbus 85]
Forbus, K. D.
The Role of Qualitative Dynamics in Naive Physics.
In Hobbs, J. R. and Moore, R. C. (editors), *Formal Theories of the Commonsense World*, pages 185-226. Ablex Pub. Co., Norwood, NJ, 1985.

[Galambos, Abelson, Black 86]
Galambos J.A., Abelson, R.P., Black, J.B.
*Knowledge Structures*.
Lawrence Erlbaum, 1986.

[Grosz, Appelt, Martin, and Pereira 87]
Grosz, B.J., Appelt, D.E., Martin, P.A., and Pereira, C.N.
TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces.
*Artificial Intelligence* 32:173-243, 1987.

[Haas & Hendrix 83]
Haas, N. and Hendrix, G. G.
Learning by Being Told: Acquiring Knowledge for Information Management.
In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (editors), *Machine Learning, An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA, 1983.

[Hayes, 85]
Hayes, P. J.
The Second Naive Physics Manifesto.
In Hobbs, J. R. and Moore, R. C. (editors), *Formal Theories of the Commonsense World*, pages 1-36. Ablex Pub. Co., Norwood, NJ, 1985.

[Hobbs 85]
Hobbs, J.R.
Ontological Promiscuity.
In *Proceedings of the ACL-85*, pages 61-69. 1985.

[Hobbs, et al 86]
Hobbs, J.R., et al.
Commonsense Metaphysics and Lexical Semantics.
In *Proceedings of the ACL-86*, pages 231-240. 1986.

[Kaczmarek, Bates, & Robins 86]
Kaczmarek, T.S., Bates, R., and Robins, G.
Recent Developments in NIKL.
In *Proceedings AAAI-86*. Philadelphia, PA, 1986.

[Larkin, Reif, and Carbonell 86]
Larkin, J.H., Reif, F., and Carbonell, J.G.
FERMI: A Flexible Expert Reasoner with Multi-Domain Inference.
*Cognitive Science* 9, 1986.

[Lebowitz 85] Lebowitz, M.
Researcher: An Experimental Intelligent Information System.
In *Proceedings IJCAI-85.* Los Angeles, CA, 1985.

[Lenat, Prakash, & Shepherd 85]
Lenat, D., Prakash, M., Shepherd, M.
CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks.
*AI Magazine* VI:65-85, 1985.

[Levesque & Brachman 85]
Levesque, H.J., and Brachman, R. J.
A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version.
In R.J. Brachman and H.L. Levesque (editor), *Readings in Knowledge Representation.* Morgan Kaufmann, 1985.

[Mauldin 85] Mauldin, M.L.
Better Information Retrieval Through Linguistic Sophistication.
In *Proceedings of the International Conference on Artificial Intelligence*, pages ?. 1985.

[Monarch 87] Monarch, I.A.
Knowledge and Hypertext: Frames and Productions.
In *submitted to Hypertext '87.* 87.

[Monarch & Carbonell 87]
Monarch, I.A. and Carbonell, J.G.
CoalSORT: A Knowledge-Based Interface.
*IEEEExpert* 2(1):39-53, Spring, 1987.

[Mooney and DeJong 85]
Mooney, R. and DeJong, G.
Learning Schemata for Natural Language Processing.
In *Proceedings Ninth IJCAI*, pages 681-687. 1985.

[Nirenburg & Raskin 87]
Nirenburg, S. and Raskin, V.
The Analysis Lexicon and the Lexicon Management System.
In *Proceedings of the ACL-87 - to appear.* 1987.

[Tou, Williams, Fikes, Henderson, and Malone 82]
Tou, F.N., Williams, M.D., Fikes, R., Henderson, A., Malone, T.
RABBIT: An Intelligent Database Assistant.
In *Proceedings of the National Conference on Artificial Intelligence*, pages 314-318. 1982.

[Tucker, Nirenburg, & Raskin 86]
Tucker, A.B., Nirenburg, S., and Raskin, V.
Discourse and Cohesion.
In *Proceedings of COLING-86*, pages 181-183. 1986.

## List of Figures

Figure 4-1: ONTOS Screen - Inserting ARTIFACT 8
Figure 4-2: ONTOS Takes The Initiative - Adding MAKES and HUMAN 9
Figure 4-3: Human Making Can Involve Plans 10

# The KADS System

*Anjo Anjewierden*

Department of Social Science Informatics
University of Amsterdam
Herengracht 196
1016 BS Amsterdam
The Netherlands

anjo@swivax.uucp

## 1. Introduction

Knowledge engineering tools are (software) tools which aid the knowledge engineer (KE) during one or more phases of the development of a knowledge based system (KBS). Most commercially available knowledge engineering tools (e.g. expert system shells, AI toolkits like ART, KEE and Knowledge-Craft) only help the KE in actually implementing a KBS. These tools provide no help in the analysis phase of KBS development (feasibility, knowledge acquisition, requirements analysis).

In this paper we will look at some existing tools which help KEs during knowledge acquisition and at the KADS system in particular.

## 2. Knowledge Acquisition

Lacking a suitable definition of knowledge acquisition we use a description adapted from Wielinga et al. (1986; p.1):

> *"Knowledge acquisition consists of the elicitation and interpretation of data on the functioning of expertise in some domain, in order to design, build, extend, adapt or modify a knowledge based (expert) system (KBS). In this view, knowledge acquisition is a permanent and crucial activity throughout all stages of designing, implementing and maintaining an expert system."*

Knowledge acquisition is an acknowledged bottleneck in KBS development and AI has very little to say about methods or techniques that could be used to alleviate it. The same goes for tools. During elicitation the KE is typically equipped with a tape-recorder and a generous supply of batteries, and during identification and interpretation pencil and paper are the tools one will find on a KE's desk.

Recent research has taken up the challenge to develop methodologies (or theories) which give the KE some guidance while developing a KBS. The KADS methodology developed in ESPRIT P1098 (Breuker, 1987; Barthelemy et al., 1987), proposes KBS development to be guided by a modelling approach and a KBS life-cycle. DETEKTR by Tektronix, (Freiling et al., 1985) gives guidelines based on a step-by-step approach. Both methodologies stress the importance of knowledge acquisition and the identification and interpretation of knowledge before prototypes are implemented.

KADS and DETEKTR also provide a handle on what knowledge acquisition support tools should do and look like. Both projects have produced tools which partly support the methodologies. The class of domains addressed by DETEKTR is restricted ("trouble shooting electronic devices") and so are the

† The work reported here is funded by the CEC as part of ESPRIT Project 1098. Many thanks to Bob Wielinga and Joost Breuker for comments on earlier drafts, to John Allen, Philip Hesketh and Yorick Phoenix for implementing parts of the KADS system, and to STC/ICL for providing some of the material used in the examples.

supporting tools. KEATS, developed by British Telecom and the Open University (Motta et al., 1986), possibly fits in the methodology/tools category as well. The KEATS report, however, only clarifies what the toolkit does, a supporting theory is only mentioned in passing. KEATS may be a typical example of an AI system, where the theory and the system are one and the same.

All three are real toolkits in the sense that they do what one could reasonably expect of them. The first screwdrivers for knowledge acquisition.

### 2.1. Levels in knowledge Acquisition

The role that knowledge acquisition plays in KBS development is given in Breuker (1987; p. 8):

> *"Experts do not express their knowledge and know-how in a rule or frame like formalism, nor are verbal data easily mapped upon such formalisms. Strategies and inference methods can only be abstracted if such categories as goals, premises, facts etc. can be read from the data. For knowledge acquisition some special purpose formalism at an intermediate level is required. (...) Brachman (1979) identifies primitives at five different levels of representation of static knowledge. These levels are: linguistic, conceptual, epistemological, logical and implementational. These levels can be extended to the analysis of verbal data on expertise..."*

Knowledge acquisition typically proceeds by repeatedly cycling through the following steps: creating an interview plan (who to interview, topics to address, interview technique to use), the interview itself (with expert, organisation or users of the KBS), and analysis of the interview. Figure 1 shows how the KE may perceive the knowledge acquisition process at a particular point in time. The figure is intentionally chaotic.

At the bottom is the linguistic level, where the KE identifies knowledge. This knowledge is available in protocols (shown as large rectangles) obtained from interviewing experts. The grey lines (threads) between the protocols indicate the structure imposed by the KE and the small rectangles indicate the observations and annotations made.

At the knowledge conceptualisation level relevant concepts, relations between concepts and structures of concepts, are represented. The threads from the protocols indicate where the information came from. For example, if the domain is about diagnosing faults in IC-boards, the concepts include the components of such boards, functions of components and relations such as *consists-of*, *causes* and perhaps *connects*.

At the epistemological level the elements of expertise are structured in an epistemological framework. The framework describes how the knowledge identified and conceptualised at the two lower levels fit in a "conceptual model" for the domain. The "conceptual model" can be viewed as an abstract description of how the expert solves a problem and the types of knowledge needed to solve that problem. Generic models, applicable to a class of domains, (called *interpretation* models in KADS) have been identified (e.g. heuristic classification, Clancey 1985).

The logical level, where the elements in the framework are mapped onto AI formalisms and techniques, and the implementational level are omitted here, as they are catered for by AI toolkits. These two levels also have threads to the levels below.

The figure and the distinction of levels tries to capture the nature of knowledge acquisition. It shows that knowledge acquisition is difficult to formalise as the KEs view of the expertise continuously changes and at each level the KE may need to go back to lower levels to remedy gaps in the data or restructure the threads. This view corresponds to the role knowledge acquisition plays in the description given earlier.

One can use the levels in two principal ways, the first is to work upwards, ultimately arriving at the "conceptual model" and use that to guide the design and implementation of the KBS. If a generic "conceptual model" is available, perhaps because a KBS has been built for a similar domain, the model-driven approach is probably faster because the KE knows what to look for (i.e. which types of concepts and relations are important, how to obtain domain knowledge, etc.). In practice, the development of a

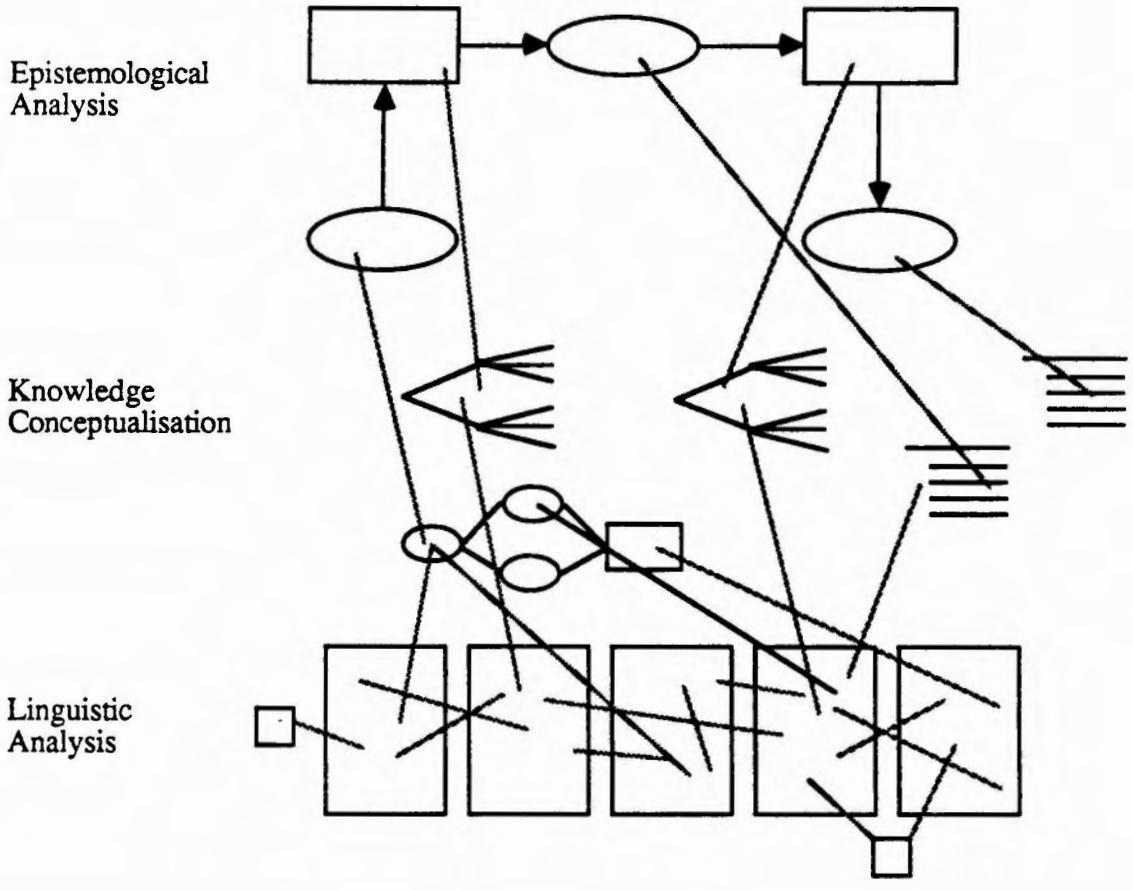



**Figure 1. Levels in knowledge acquisition**

KBS will often be a mixture of the two approaches.

We will use the three levels in the remainder of this paper to illustrate what existing knowledge acquisition tools support. The KADS methodology (Breuker, 1987), which is partly built around these levels, and the KADS system described in the next section will give more detail.

### 2.2. Existing Tools

Some of the information in this chapter was obtained from the proceedings of a workshop on knowledge acquisition, held in Banff (Canada) in November 1986. The proceedings reflect current research on knowledge acquisition.

Very few tools deal with the linguistic level (protocol analysis). KADS and KEATS support the linguistic level by offering a tool which allows the KE to structure the information in the protocols. Research at Boeing focuses on merging interviewing and protocol analysis by giving the expert the means to enter the knowledge directly into an Expertise Transfer System (Boose, 1986). An extension to ETS, called Aquinas, formalises the knowledge entered and can be used as an intelligent front-end to AI toolkits (Boose & Bradshaw, 1986). Aquinas is an example of a system where the entire KBS

development process is automated, potentially without even using KEs. The weakness of Aquinas is the use of particular knowledge elicitation technique ("rating grids"), which is thought to be insufficient for general KBS development. For those who wonder, Boeing uses the system to produce Personality Disorder and Wine Advisors amongst others, they do not use it to design aircraft.

DETEKTR (Freiling et al., 1985) uses an interesting approach to the linguistic and conceptualisation levels. The methodology first tells the KE to create a paper knowledge base, i.e. a collection of knowledge obtained from interviewing electronic engineers. The paper knowledge base is in English, and the next step is to formalise the knowledge base into a knowledge acquisition grammar which still resembles English. The advantage here is that the grammar can both be understood by the electronic engineers (for communication and validating expertise) and automatically transformed into knowledge representation formalisms. The DETEKTR methodology also follows the five levels given above.

The conceptualisation level is indirectly supported by KREME (Knowledge Representation Editing and Modeling Environment) developed at BBN (Abrett & Burstein) and by KEATS. They approach the problem of knowledge acquisition (and knowledge representation) by offering the KE a wide range of representational formalisms, incremental creation of the knowledge base, classification and consistency checking. Although KREME and KEATS are more oriented towards the logical and implementational levels of KBS development, the richness of the environments and the use of direct manipulation of graphical representations make it possible to use them as a tool for the conceptualisation level as well. This is generally not true of AI toolkits where the user is bound to a single or sometimes a few formalisms in which to express knowledge. KEE, for instance, only supports isa relations, while *consists-of* or causal relations are difficult to represent without much programming effort.

Systems which work downwards from a (generic) "conceptual model" have also been built. ROGET (Bennett, 1985) is a generalisation from MYCIN and helps the KE by determining a domain specific conceptual structure by asking questions about the domain, and giving examples from the library of conceptual structures it knows about. ROGET is limited in the sense that it can only cope with domains similar to MYCIN.

Most of the tools described above are experimental and used for further research into various aspects of knowledge acquisition. Remarkably, none of the Banff papers explicitly states the desire to develop tools which ease the KE's job by offering the knowledge acquisition equivalent of a word-processor. If knowledge engineering is to become a viable enterprise undertaken by people who are now using conventional and predictable tools, this is the first hurdle to take.

## 3. KADS

As part of the ESPRIT P1098 activities aimed at developing a methodology for KBS development, several versions of a knowledge acquisition support system, KADS, were developed. The first version of KADS was conceived as a top-down implementation of the KADS methodology, aiming to demonstrate the applicability of the methodology by implementing it as a tool. Although KADS was interesting, experiments within ESPRIT Project 1098 quickly revealed that it was not of much practical use. Apart from problems with the robustness of the system, the main reason why the first KADS system failed to serve its purpose was the lack of flexibility both in the user interface and the normative component. Knowledge acquisition is a difficult and complex process, which can only be supported by a highly flexible system.

The second version of the KADS system separates bottom-up tools, and the normative, top-down component in two subsystems, which will ultimately be integrated into one system. KADS Power Tools provides bottom-up support for knowledge acquisition by offering a set of tools, of which some are general and could be used outside the scope of knowledge acquisition, and some are tied to the methodology. KADSVENTURE is the normative part of KADS. It provides guidance and knowledge-based support to the methodology. A prototype of KADS Power Tools (hence KPT) is now operational and used within the project, KADSVENTURE is currently being designed.

The minimal requirements for KPT were obtained through meta-knowledge acquisition, i.e. interviewing KEs on how they solved their problems with knowledge acquisition. Apart from functional

requirements, it was clear that KPT should have a pleasant user interface. Unfortunately, a description on paper of the system cannot convey the "feel" of the user interface. Most operations in KPT are done by pointing and clicking or dragging the mouse.

### 3.1. Environment

KPT is designed around state-of-the-art software engineering and AI techniques. It is written in a hybrid programming environment called PCE-Prolog (Anjewierden, 1986), consisting of PCE, an object-oriented programming system and Quintus Prolog. PCE, written in the C language, was developed in parallel with the original KADS system and features high-level windowing, graphics and text-manipulation facilities. The programmer views PCE-Prolog as a whole. All PCE related facilities are available through four Prolog predicates. PCE-Prolog runs on Sun Workstations and contributed substantially to the short development cycle of KPT.

### 3.2. Protocol Editor

PED, the KADS Power Tools Protocol Editor, supports the linguistic or knowledge identification level in knowledge acquisition. PED is basically a dedicated hypertext system. Hypertext (Nelson, 1980) is the idea that information should be accessible in a non-sequential way. For instance, someone reading a paper should be able to select a word and see its definition, follow references or read other people's comments. Some hypertext systems have recently become widely available (Guide on the Apple Macintosh for instance), but there is still little agreement on what a general purpose hypertext system should provide and, perhaps more important, how it should be provided.

In PED, hypertext is the technical solution to the problem of dealing with related written materials. Every hypertext system defines a set of link types between text. Nelson, for instance, defines *jump-*, *quote-*, *note-* and *equi-*links, which roughly corresponds to how people want to read or write. The situation is different for protocol analysis. The KE does not need to write a document, but s/he must study the protocols and identify knowledge for later use and topics and questions for further elicitation sessions.

The first principle of PED is that the protocols, as transcribed from interviewing sessions with the expert, should not be modified. When the KE starts modifying rather than annotating protocols, some vital data, which seem irrelevant at the time may be lost. In PED this principle is translated by imposing a structure *on* the protocol and not *inside* it. The second principle is that a team-effort must be supported. Most other hypertext systems provide facilities fo each user to create a personal view of the information available. In a team of knowledge engineers multiple views may be required simultaneously. This requirement is met through visualising the links created by the KE explicitly, rather than showing them on demand, and providing a facility to print arbitrary sections of the protocols based on the links they have.

All link-types in PED are defined on fragments. A fragment is a contiguous user specified part of a text (i.e. a protocol). Fragments are superimposed on the text and all changes to the text are therefore mediated to the fragment. Obviously, fragments may have any length, they could refer to a word, sentence, paragraph or even the entire protocol.

The following table shows the PED links between types of objects (links marked with * require a KE specified name, all links are bi-directional):

| *Link type* | *From <=> To* |
|---|---|
| annotation | Fragment <=> Text or Diagram |
| group | Fragment <=> Group |
| link | Fragment <=>* Fragment |
| concept-link | Fragment <=>* Concept |

An *annotation* is similar to a footnote in a document. Annotations are used to make observations about a fragment, for instance to point out lack of knowledge or to describe the KE's understanding of what the expert says. It is up to the KE how s/he annotates a protocol.

A *group* is a collection of fragments. Groups allow the KE to structure protocols by giving a name to a collection of related fragments. The relation between the fragments may be superficial ("about-user"), about a domain concept ("micro-processor") or may be some high-level abstraction ("inferences"). The important point is that grouping allows the KE to create collections of fragments which s/he thinks are related. Syntactic approaches to grouping of fragments, e.g. keyword matching, are necessarily more restrictive. Any fragment may be a member of any number of groups.

A *link* is a link between two fragments, the link is given a name. Named links provide a general solution to the hypertext problem of how many link-types should be provided. PED's answer is one, but the KE should be allowed to give a name to that link. Named links allow the KE to create "contrast", "detail" and "continues" (etc.) links between fragments, thereby once again offering a facility to impose a meaning on how the fragments are related. Named links in PED can be used to emulate every link-type anybody ever comes up with.

A *concept-link* is a link between a fragment and a concept. Concepts typically refer to domain concepts, and the name of the concept-link can be used to describe attributes or give a definition. For instance, if the KE sees a definition of a concept, s/he can link the fragment which contains that definition with a "definition" link to the concept. Or if the expert talks about the voltage property of transistor-2, a "voltage" link can be made.

The implementation of links is based on the ability to *hyper* any object to any other object in PCE. PED has no restrictions, the KE can load any number of protocols and create any number of fragments, notes and links.

The PED user interface has an interesting feature which will hopefully solve some of the potential user interface problems associated with hypertext systems. A protocol is displayed in a protocol window (large window in figure 2). A protocol window consists of a text part (on the left) and a margin (on the right). The margin contains a marker, shown as a small icon, for each link in the visible part of the protocol. The text part and the margin scroll simultaneously, the markers therefore always point to fragments in the text shown.

The protocol window clearly indicates that there are links in a given portion of the protocol and also identifies what type of object these links refer to. Analysing protocols in PED is, therefore, like writing notes-in-the-margin as the KE would do when confined to paper.

A KE reading a protocol entered into PED uses the markers both for obtaining precisely what the marker represents and for jumping to the other end of the link. An alternative representation of the links can be displayed in a web. A complete web is obviously useless if the number of links is large, so the KE has to choose a starting node (i.e. a fragment, group or concept) and PED displays a web of all links from that node, the web can obviously be expanded from any given node.

Against the hypertext philosophy, PED is willing to print arbitrary selections from the protocols. The KE specifies a search criterion, for instance all links to a particular concept or all members of a group, and PED then collects, formats and prints the fragments found. These print-outs can be used to communicate with the expert ("this is what I know about transistor-2"), for further study and for documentation purposes.

The only system we know of that also supports protocol analysis by exploiting the hypertext idea is CREF in KEATS (Motta et al., 1986; pp. 6-9). CREF places more emphasis on manipulating collections of fragments and predefines the links fragments may have.

### 3.3. Concept Editor

CE, the KPT Concept Editor supports the knowledge conceptualisation level in knowledge acquisition. At this level knowledge is organised as concepts, relations and structures of concepts over a certain relation.

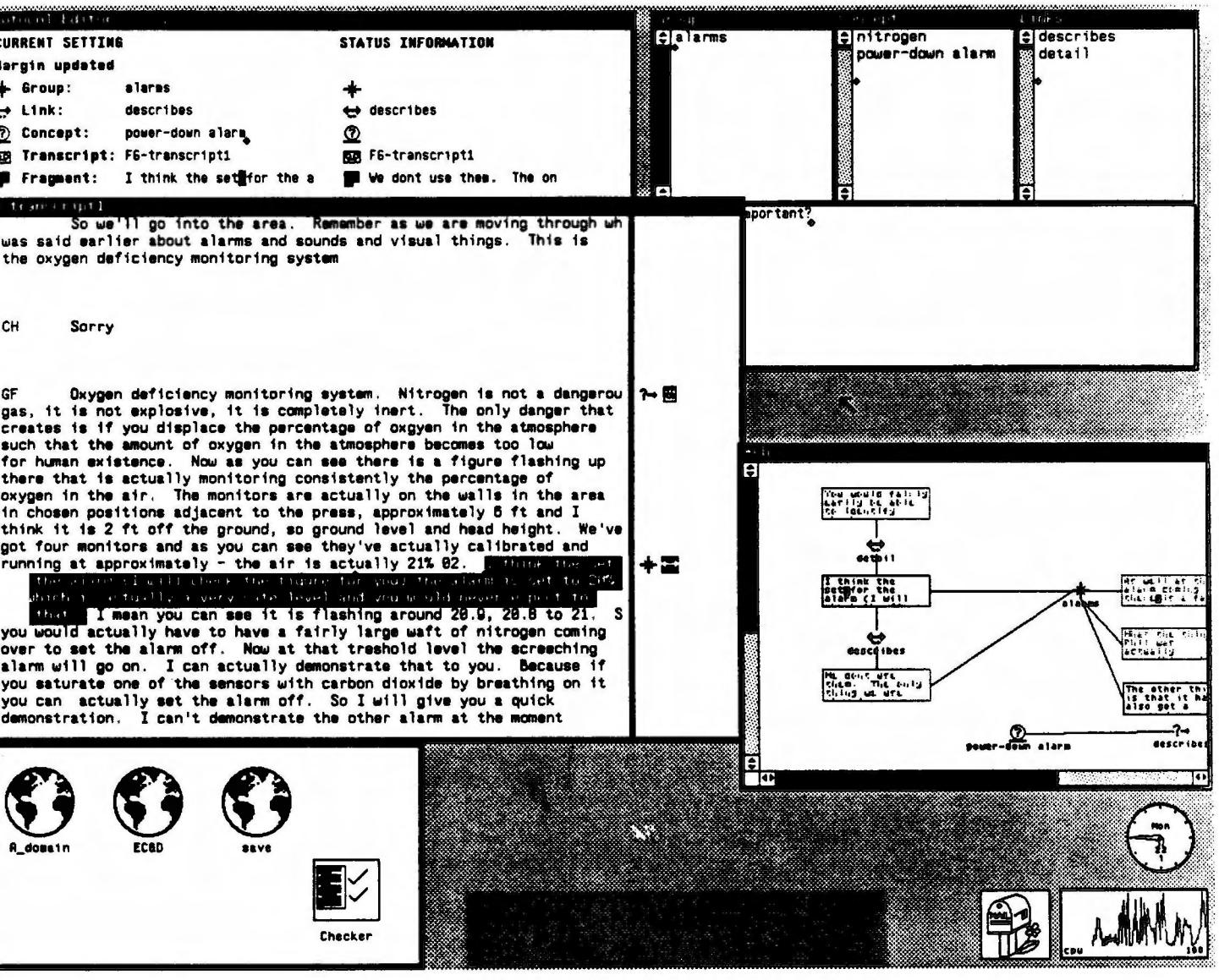



**Figure 2. Protocol Editor**

The Concept Editor knows that concepts may have attributes (or properties), and that these attributes may have a value. It also knows that concepts may be related to other concepts. Precisely what attributes and relations are used highly depends on the domain. For instance, in a domain related to car manufacturing, attributes like the materials of which a car is made are important, whereas in the domain of selling cars attributes like cost, fuel consumption and price are relevant. It is up to the KE which attributes are important for the domain at hand, there is no obligation to completely define these concepts. Indeed, for most concepts this is impossible. Concept attributes are given as pieces of text, so a list of attributes for a car could be as follows:

All attributes for concept **car** in domain **transport**

| | |
|---|---|
| *carries* | people |
| *colour* | a colour |
| *consumes* | petrol |
| *made-by* | a car-manufacturer |
| *synonyms* | automobile |

The *is-a* relation plays an important role in many representations at the implementation level (e.g. in object-oriented and knowledge representation systems). In the transport domain, one would perhaps find the following isa hierarchy of concepts:

**isa** from **vehicle** in domain **transport**

vehicle
- aircraft
- bike
- boat
- car
  - bus
  - taxi
  - truck

At the conceptualisation level other relations between concepts may also be important, e.g. *consists-of*, causal or temporal relations. A simple *consists-of* hierarchy for a car looks like this:

**consists-of** from **car** in domain **transport**

car
- wheel
- seat
- chassis
- engine

Which hierarchies (relations between concepts) are relevant depends on the domain, in some domains it may be sufficient to define "wheel" as an attribute of a car, whereas in others it could be important to specify what a "wheel" is made of and how wheels relate to other parts of a car.

The above examples indicate the purpose of and some of the requirements for the Concept Editor. The KE needs a tool which allows incremental definition of concepts and the relations they have. Since the KE, at this stage, is still trying to understand what knowledge is important in the domain, the tool should be flexible and preferably not complain about possible inconsistencies other than syntactic errors such as a concept that is -indirectly- a sub-concept of itself.

Documentation is considered important in all (KBS) methodologies. To quote Freiling et al. (1985; p. 158):

> *"All too often, knowledge engineering projects become a black hole, and managers have difficulty perceiving signs of progress. With a clear sense of stages and documentation which can be delivered at each milestone, it becomes possible to say "We've completed the knowledge organisation phase and we're now defining the representation," rather than "We're working on it.""*.

The Concept Editor, therefore, can format and print the information that has been entered into the system. Once again, these documents can also be used to verify with the expert whether the KE's understanding of a particular aspect of the domain is correct.

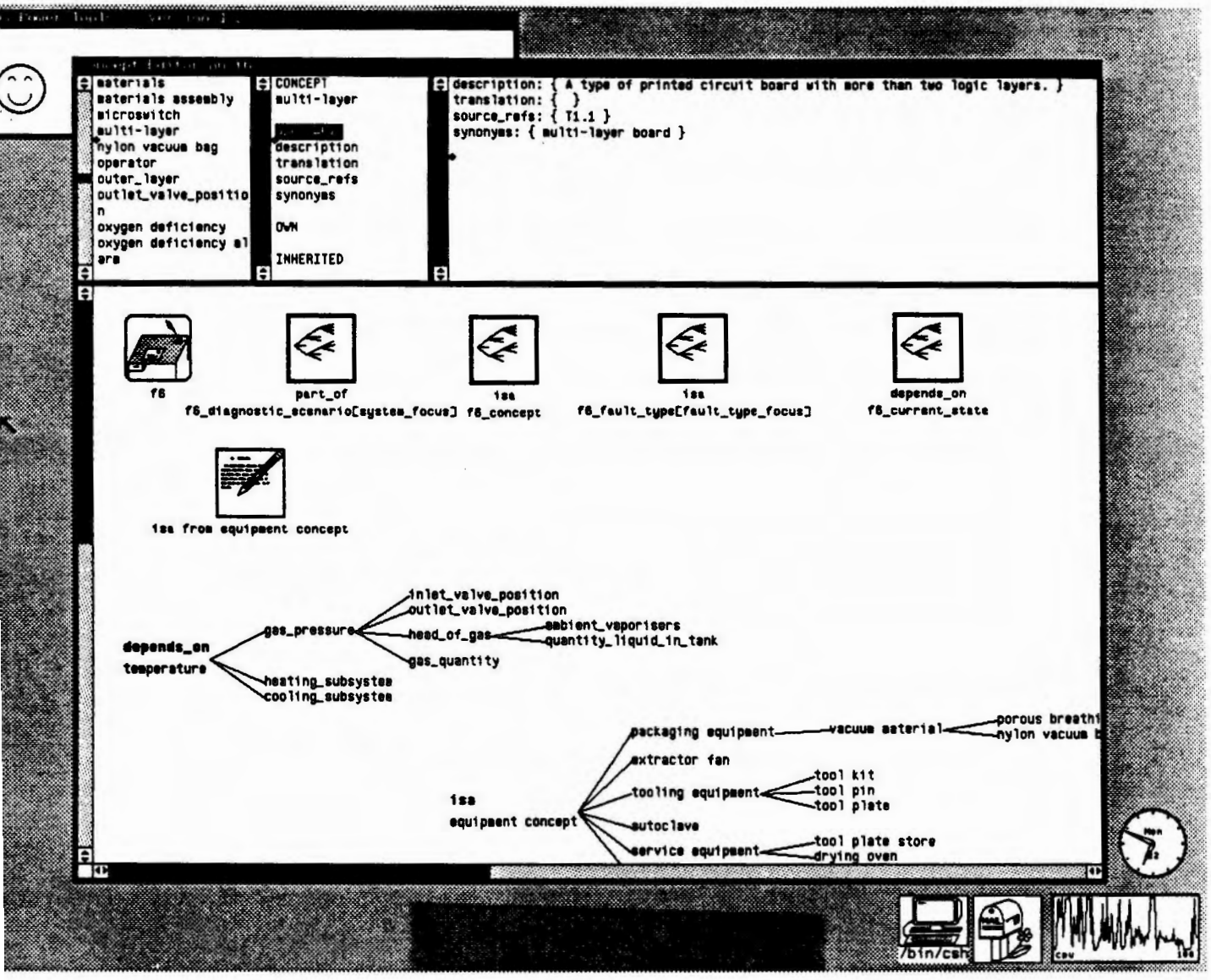



**Figure 3. Concept Editor**

The Concept Editor is shown in action in figure 3. From left to right and top to bottom the sub-windows contain: a list of all concepts in the domain, a list of all attributes of a particular concept, a text window and a graphics window known as the desktop. The text window is used to show the concept attributes and other text the KE wants to edit or view. The desktop is the main means of interaction. The user interface is a mixture between the Smalltalk style (paned windows and popup menus) and the Macintosh style (objects represented by icons, clicking and typing as the basic mechanisms).

CE allows the KE to create concepts and hierarchies of concepts over a given relation. CE knows about one relation which is provided by default, *isa*. If two concepts have an *isa* relation, attributes are inherited. Other relations have no semantics at all. The KE creates new relations by typing the name of the relation, thereby offering the possibility of relating the concepts in a way that is most suitable for

the domain at hand.

The figure shows the *isa*, *part-of* and *depends-on* relations identified useful in a domain called "diagnosis in a bonding shop". Each hierarchy only shows the concepts over a given relation from a given root-concept, implying that a single concept can be in more than one hierarchy and that more than one disconnected hierarchy over the same relation may exist.

CE has implicit support for building a glossary or lexicon of domain concepts. Each concept has four attributes (shown in the figure) which are initially empty and can be filled by the KE.

### 3.4. Interpretation Model Librarian

The IML (Interpretation Model Librarian) and the Conceptual Model Editor, which is not described here, support the epistemological level in knowledge acquisition. Interpretation Models (IMs) are specific to the KADS methodology as described above (see Wielinga & Breuker (1986) and Breuker & Wielinga (1987) for details).

The primary function of IML is to allow the KE to choose one or more IMs that fit the domain. Like in an ordinary library, no new IMs may be added by the user although the librarian (a person) can add or modify models.

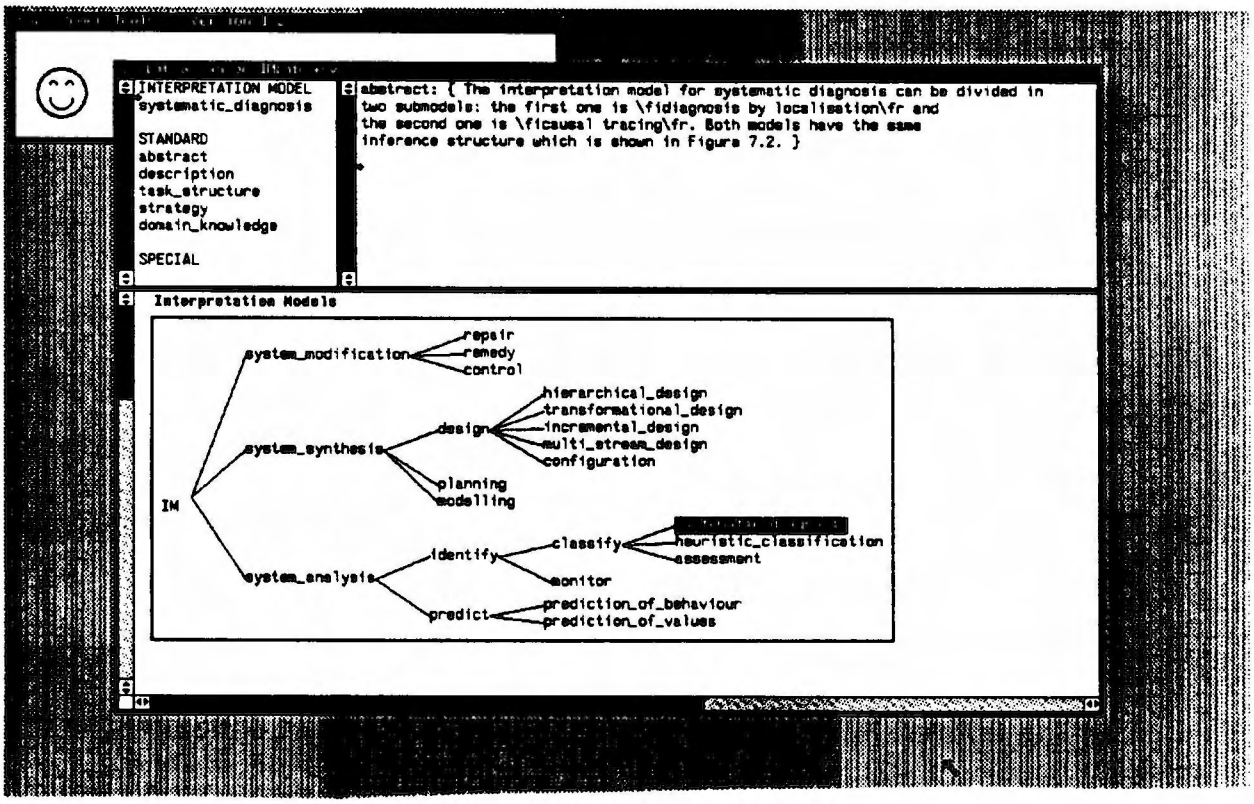



**Figure 4. Interpretation Models available in the Librarian**

After entering the IML, an isa hierarchy of IMs is shown (see figure 4). Clicking on a model in the hierarchy shows its attributes and puts up the abstract. Additional information about the model can be obtained.

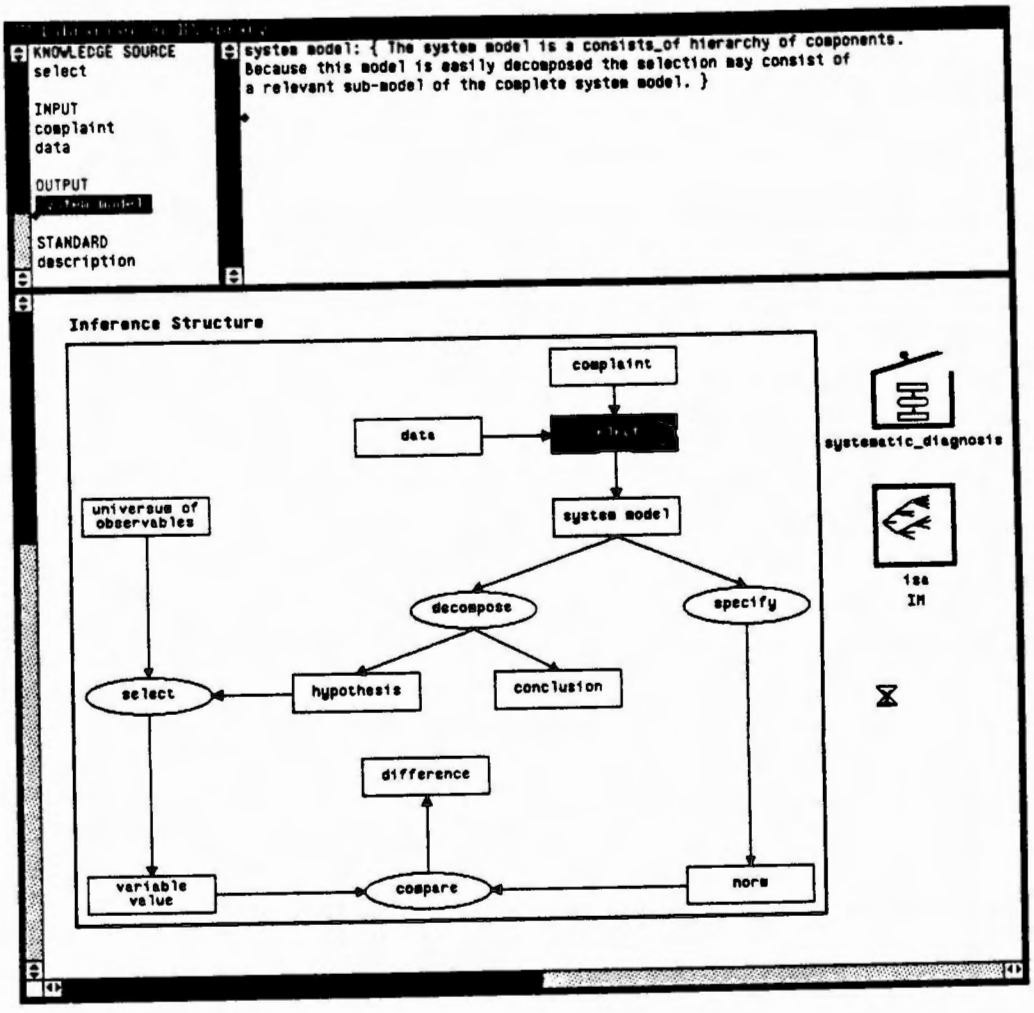



**Figure 5. Part of "Systematic Diagnosis"**

Figure 5 shows the inference structure for the IM called "systematic diagnosis". Inference structures describe one part of an IM, and consist of knowledge sources (shown as ellipses in the figure) and meta-classes (shown as boxes). Knowledge sources are functional descriptions of inference making, and meta-classes serve as the input and output parameters. Meta-classes map onto structures of domain concepts (often the hierarchies created in CE). Typically, a knowledge source is an abstract description of how its input is converted to its output.

Knowledge sources and meta-classes also have attributes. The figure shows the "select" knowledge source and what role the "system model" meta-class plays for that knowledge source. The input and output meta-classes for an inference structure are dynamically determined from the diagram. If the librarian would add another meta-class to one of the knowledge sources, the attributes are updated.

IML further allows the KE to print an IM, in which case all information is retrieved and printed in a standard format, and to choose an IM for the domain. Choosing an IM causes it to be copied into the domain and the KE can then map the meta-classes onto hierarchies of domain concepts.

## 4. Conclusions

We feel that the *Power Tools* approach to support knowledge acquisition can enhance the activities undertaken by the KE. A substantial amount of time during the design of KPT was spent on defining the user interface. KPT now draws a large part of its acceptance from that user interface.

The tools described here do not help the KE deciding which activities to undertake and how these activities relate to each other. The only guidance the KE gets is by a paper methodology, future extensions of KADS, notably KADSVENTURE, will address these issues.

## 5. References

Abrett, G. and Burstein, M. (1986) The KREME Knowledge Editing Environment, *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Anjewierden, A. (1986) *PCE-Prolog 1.0β Reference Manual*, ESPRIT Project 1098, University of Amsterdam, (revised version in preparation).

Anjewierden, A., Allen, J. and Hesketh, P. (1987) *KADS Power Tools User Guide*, ESPRIT Project 1098, University of Amsterdam and STL.

Barthelemy, S., Edin, G., Toutain, E. and Becker, S. (1987) *Requirements Analysis in KBS Development*, ESPRIT Project 1098, Deliverable D3, Cap Sogeti Innovation and SCS.

Bennett, J., (1985) ROGET: a Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System, *Journal of Automated Reasoning*, 1, pp. 49-74.

Brachman, R. (1979) On the Epistemological Status of Semantic Networks, in: N.V. Findler (ed.) *Associative Networks*, Academic Press, New York.

Boose, J. (1986) *Expertise Transfer for Expert System Design*, Elsevier, New York.

Boose, J. and Bradshaw, J. (1986) Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge Acquisition Workbench for Expert Systems", *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Breuker, J.A. (ed.) (1987) *Model-Driven Knowledge Acquisition*, ESPRIT Project 1098, Deliverable A1, University of Amsterdam and STL.

Breuker, J.A. and Wielinga, B.J. (1987) "Knowledge Acquisition as Modelling Expertise: The KADS Methodology", *Proceedings First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading (UK).

Clancey, W. (1985) Heuristic Classification", *Artificial Intelligence*, 27, pp. 215-251.

Freiling, M., Alexander, J., Messick, S., Rehfuss, S. and Shulman, S. (1985) Starting a Knowledge Engineering Project: A Step-by-Step Approach", *AI Magazine*, (3).

Motta, E., Eisenstadt, M., West, M., Pitman, K. and Evertsz, R. (1986) *KEATS: The Knowledge Engineer's Assistant*, Human Cognition Research Laboratory, The Open University.

Nelson, T. (1980) Replacing the Printed Word: A Complete Literary System, *Information Processing 80*, IFIP, North-Holland, pp. 1013-1023.

Wielinga, B.J. and Breuker, J.A. (1986) Models of Expertise", *Proceedings of ECAI 1986*, Brighton.

Wielinga, B.J, Breuker, J.A. and van Someren M.W. (1986) *The KADS System: Functional Description*, ESPRIT Project 1098, Deliverable T1.1, University of Amsterdam.

# KRIMB: An Intelligent Knowledge Acquisition and Representation Program for Interactive Model Building

Louis Anthony Cox, Ph.D. and Richard Blumenthal

U S WEST Advanced Technologies
6200 South Quebec St.
Englewood, CO 80111
(303) 889-6025

August, 1987

## 1. Introduction

Many paradigms for automated machine learning and knowledge acquisition have been proposed recently, including learning by induction [La 86, Qu 86] or generalization [Mi 83], learning by example/explanation, learning by doing, and learning by being told [Ha 83]. This paper describes an experimental system called KRIMB being developed at US West Advanced Technologies to further research on knowledge acquisition by intelligent questioning of the user [Sa 86, Bo 85]. One of the earliest and still one of the most successful paradigms, intelligent questioning requires the system to use general knowledge of an application domain to help gain knowledge about specific systems or situations. General knowledge is used both to frame appropriate questions and to guide the dialogue so as to elicit useful facts efficiently from a user or domain expert.

In KRIMB, unlike most earlier diagnostic systems familiar in the expert systems literature, knowledge acquired from the user is used to build highly structured operations research models of complex systems to aid in tasks such as fault prediction. For example, models built by KRIMB can be used for reliability analysis using conventional algorithms and software. The process of building a model on the basis of a dialogue with the user, however, requires artificial intelligence techniques.

Because the application domains for KRIMB (e.g., complex engineering systems) are exceptionally rich and structured, efficient knowledge acquisition requires some new techniques and maximum use of information provided during the dialogue with the user. As an example, supplementing acquired facts with conjectured generalizations that are assigned a status of "possible"

rather than a conventional truth-value reduces the number of questions that need to be explicitly asked without compromising the soundness of the models built by the system or the soundness of inferences based on them. This paper outlines the principal knowledge acquisition techniques used in KRIMB to expedite modeling of complex systems.

## 2. Background

KRIMB was originally developed to demonstrate some points about the relations among knowledge acquisition, knowledge representation, and purposive model building. Initially, it was designed to acquire knowledge and to reason about the reliabilities of complex engineering systems, such as data communication networks or nuclear power plants. However, it is being extended to support reasoning about general flows in complex systems, including, for example, flows of interoffice mail in large organizations.

The design of KRIMB incorporates several working assumptions about the nature of good knowledge acquisition. It is based on the following somewhat controversial hypotheses:

1. A "good" knowledge acquisition program should minimize cognitive strain on the user [Ca 83]. In KRIMB, this goal is sought by leaving to the program the tasks of taking the initiative (unless interrupted) in structuring the knowledge acquisition interview, and in formulating its requests for information as questions requiring only one-word (or menu-driven) answers. However, the user is allowed to override the questioning sequence at any time to interject a query or an assertion, or to refocus the dialogue in a desired direction.

2. A "good" knowledge acquisition program should be goal-directed.

Work in discourse understanding has demonstrated the importance of goals in focusing attention [Gr 85]. In KRIMB, a primary goal of the knowledge acquisition interview is to enable the program to build a model of the system that the user is describing, and to populate the model with the data it needs to perform reliability calculations and risk analyses. This model is shared with the user as it is being built, through a graphic interface that shows a model as a multi-level network. The model can be edited and refined by the user if and when he wishes to add or change details. The version of the model that the user currently approves is used to perform calculations and to answer user queries.

3. A "good" knowledge acquisition program requires that the knowledge acquisition and knowledge representation processes be integrated [Co 87]. Instead of first choosing an application and then seeking a suitable representation for it, as has been the norm among knowledge engineering practitioners, our approach is to define a knowledge representation system (KRIMB) that is sufficient for describing a broad class of complex systems. KRIMB assumes that the user's application can be represented in this pre-defined "ontology", and exploits this assumption to streamline knowledge acquisition. In this respect, KRIMB follows the tradition of data base models, but applies it to knowledge base modeling. This inversion of the usual relation between knowledge acquisition and knowledge representation is potentially the most controversial of the assumptions underlying the KRIMB system.

## 3. Levels of Knowledge Extraction and Representation in KRIMB

### A. Ontology and Meta-Knowledge

KRIMB supports several types of interactions with builders and users of models and modeling knowledge. At the top level, a system developer provides an ontology [Al 86, Qu 80], consisting of primitive object categories and relations between them. Logical truths and definitions of meta-predicates or meta-relations (such as the subset relation between object classes, which is denoted by a meta-predicate called "type-of") that can hold in any application domain and that are thus devoid of empirical content are also found at this level. Meta-predicates specify necessary and possible relations among object classes, and are used later in defining the "meanings" of object class names for the purposes of terminological competence, much as in KL-ONE and its descendants [Br 85, Vi 84] . Object classes and relations are defined using a frame-like notation, however, rather than via a semantic network (or a set of unbundled assertions, which was the first approach tried.) Each object class is given a (working) definition by specifying its necessary and possible features (which are themselves objects), attributes (which typically have numerical values), and relations.

At the next level down, an application domain expert fills in a meta-knowledge base for a particular application domain. The meta-knowledge base consists of (i) a set of domain-specific terms and their meanings; and (ii) a set of rules for building models from known facts about specific systems. Term meanings are defined by specialization of the object classes and meta-predicates given in the ontology [So 84]. Part of the meaning of a term is implied by the transitivity of specialization, and must be retrieved from the specialization hierarchy (a lattice)

as needed using chain-seeking procedures. While KRIMB's approach to terminological competence (i.e., understanding word meanings for the purposes of knowledge acquisition and certain types of inference) is thus similar to the approaches in [So 84, Sc 82, Si 83, Br 85 and Vi 84], KRIMB, unlike these systems, emphasizes the distinction between necessarily applicable and possibly applicable relations, features, and attributes.

As a simple example, for any animal class X there is some type of food Y such that members of X eat members of Y. The "eats" relation necessarily holds between any animal and some food. The necessity of this relation is a meta-fact about object classes (animals and foods), rather than a fact about any specific animal or food. It could be expressed through a meta-predicate such as "must-apply(eats, animal, food)." This meta-predicate might be part of an ontology for the application domain of animal dietary habits, and could be specialized for particular applications, e.g., through meta-facts such as "must-apply(eats, carnivore, meat), "must-apply(eats,tiger,meat)", etc. By contrast, a possible relation would be "eats(animal,meat)", which may or may not be true, depending on how the "animal" argument is specialized. The meta-predicate "can-apply(eats,animal,meat)" would express this meta-fact; further development of the specialization hierarchy would make the possibilities implicit in this meta-fact explicit by showing classes of animals that necessarily do eat meat. KRIMB makes use of a refined version of these distinctions (in which the existential or universal quantification of each argument is made explicit in defining types of "must-apply" and "can-apply") to control inheritance between object classes. A specialization of an object class inherits its necessary features, attributes, and relations; conversely, necessary relations for an object class become possible relations for generalizations of the class.

(Thus, from "must-apply(eats,carnivore,meat)" and "type-of(carnivore,animal)", infer "can-apply(eats,animal,meat).") These representational niceities are effective in streamlining the acquisition of knowledge about specific systems involving relations among instances of the object classes.

Note that the necessity or possibility of meta-facts is predicated of object classes, rather than of instances, reflecting a view that definitional truths can be asserted at the class level, even though empirical truths hold only at the instance level. This allows us to say that canaries fly (i.e., that must-apply(fly,canary)), and that this is (definitionally) true as a meta-statement about the object class "canary", even though there may be instances of canaries that can not fly. Also, the specification of necessary and possible aspects of an object class results in a partial definition that is sufficient for drawing certain kinds of inferences (based on entailment and subsumption among class relations.) No effort is made to define object classes by necessary and sufficient conditions, nor is such definitional completeness needed to build useful models of specific systems. (The model construction rules portion of the meta-knowledge base will be discussed later.)

Finally, "possible" applicability of a relation in the sense that we have introduced it is distinguished in KRIMB from conjectured applicability, which holds when the system posits a generalization or specialization that may not hold at all. For example, from the metafacts "must-apply(eats,animal,food)" and "type-of(rabbit,animal)" and "type-of(steak,food)," it would be possible to conjecture that "must-apply(eats,rabbit,steak)." Although it turns out to be false, this conjectured metafact is consistent with the given ones: it is possible with respect to them, although not with respect to a wider meta-fact base that

includes the knowledge that rabbits are vegetarians. This notion of *possibility with respect to what is currently known* is captured through the "may" meta-predicate, as in "may-apply(eats,rabbit,steak)." This meta-predicate can be "true" at any time only with respect to what is known.

**B. Questions, Facts, and Models**

Meta-facts about necessary and possible relations are used during a knowledge-acquisition interview to construct questions about a specific system in the application domain. For example, from the meta-knowledge "must-apply(eats,animal,food)" and the empirical fact that cat(12) is an instance of an animal, the system would construct an internal representation of the question "What type of food does cat(12) eat?" This question would be added to a pending-question stack, and eventually be asked to the user (unless the answer emerged during the dialogue before the question gets asked.) Such questions are answered (to the extent possible ) by an application expert, who provides descriptive information about the specific system to be modeled. Answers to these questions are stored as empirical facts, meaning fully instantiated meta-relations, stored in a fact base. An example of a meta-fact in a simple engineering system domain would be that (all) air conditioners require electricity flows as inputs (i.e., that the relation of requiring necessarily holds between any air conditioner and some electricity flow). An example of a fact would be that some specific air conditioner, say air conditioner(01), takes some particular flow of electricity, say flow(14), as input. The fact base stores knowledge about a particular system, while the meta-fact base contains more general knowledge that may be used in modeling many different specific systems.

Finally, known facts about a specific system are mapped into a logical model of the system by the model construction rules contained in the meta-knowledge base for the application domain. In systems reliability applications, the logical model represents failures of systems and subsystems as its basic events, and derives logical precedence constraints among these events from facts about the relations among subsystems. For example, if air conditioner(01) requires electricity flow(14) as input, then failure of electricity flow(14) would be a logically sufficient predecessor for failure of air conditioner(01). (This modeling formalism for failures is discussed in greater detail in the next section.) Once a model has been built at some level of detail and completed (by accounting for all required flows and supplying primitive failure probabilities for all subsystems that are not further decomposed within the model), appropriate reliability engineering calculation algorithms (based on "fault tree analysis") can be invoked to calculate system reliability and to identify critical failure paths for the end user. The reliability engineering details are given in [Co 86].

In summary, KRIMB represents empirical knowledge at three main levels: the level of meta-knowledge about an application domain; the level of facts about a particular system; and the level of a model of the system. It must acquire these segments of its knowledge from an application domain expert and an application expert. It uses this knowledge to interpret questions and to answer user queries with the help of appropriate numerical calculation algorithms applied to the model. The remainder of this paper focuses on the knowledge acquisition process.

## 4. Knowledge Acquisition via Intelligent Questioning

At both the meta-fact (object class) and empirical fact (instance) levels, KRIMB is designed to elicit relevant knowledge about word meanings and about specific systems through intelligent, goal-oriented questioning of an appropriate individual — the application domain expert or the specific application expert, respectively. The expert may also assert facts directly, and these will be processed and stored in the knowledge base before questioning resumes. To understand how the knowledge acquisition dialogue is generated, it is necessary to understand a little about the ontology and the model construction rules that go with it.

### A. Example Ontology for Reliability Analysis

An ontology provides fundamental "common-sense" knowledge about the way the world (or the portion of it to be represented in KRIMB 's models) works. By building in this fundamental knowledge at a high level [AI 86], we enable knowledge acquisition to take place through "intelligent" questions that incorporate knowledge of what the possible answers might be.

A useful ontology for systems reliability applications is the following:

- Object Classes: processes, resources, flows, channels

- Meta-Predicates: The following relations must apply: requires( process, resource), produces( process, resource), has( flow, resource, source process, destination process, channel). [Notation: The word outside parentheses is the name of the relation, and the words in parentheses are the names of the arguments of the relation. "Has" is a general-purpose relation for expressing features of an object class; its interpretation depends on its arguments. The

example just given should be interpreted as specifying that the source process supplies a flow of the resource to the destination process via the channel.]

**B. Model Construction Rules**

Facts about a specific system are mapped into a logical model of the system, suitable for reliability analysis, by special model construction rules. An example of such a rule is *oe(m(fails, process(X)), m(fails, process(Y))) if has(flow(Z), resource(R), source process(Y), destination process(X), channel(C)).*

This asserts that the "module" (model component) representing failure of process X "or-encloses" (abbreviated oe) the module representing failure of process Y — meaning that a sufficient precondition for failure of X is failure of Y — if process Y supplies some resource (represented by the free variable R) to process X via some flow along some channel. Such a rule would be appropriate in an application domain where all flows are necessary; it would have to be modified for an application domain that allowed for the possibility of redundant flows. A reliability model of a system consists of a network showing precedence constraints (represented by arcs) between modules (represented by nodes, and representing subsystem failures.)

Each module may contain [i.e., "and-enclose" (ae) or "or-enclose" (oe) ] other modules as substructure. The substructure of a module representing failure of a subsystem shows preconditions for this failure event -- either necessary (ae) or sufficient (oe). Modules that have no substructure must be assigned primitive failure probabilities. At any point during a dialogue in which a model is being built, the user can inspect the model visually using a graphics interface that shows modules as boxes, flows as lines connecting boxes, module failure

probabilities as box contents, and that allows "panning" and "zooming" to give access to or to hide substructure of a module. This representation scheme, which is based on an outgrowth of fault tree analysis in reliability analysis, is discussed and illustrated in detail in [Co 86].

## C. Acquiring Knowledge

An ontology allows the KRIMB program to acquire terminological and factual knowledge. It does so by generating questions that seek to *specialize* and *instantiate* known meta-facts, respectively. The goal is to link new words into the existing specialization hierarchy, which is grounded in the ontology, and to fully instantiate known applicable relations with the instance labels of specific objects to which they apply. For example, KRIMB might ask the following question during the course of a dialogue: "WHAT RESOURCE FLOWS DOES ENGIN(01) REQUIRE?" The user might reply with a list, like "gasoline(02), oil(03)," which refer to specific (hence numbered) flows of gasoline and oil, respectively. (An unnumbered version of this exchange might take place during acquisition of meta-knowledge about the application domain of car engine.) From this exchange, KRIMB observes that "gasoline" and "oil" — two words that it may not have known about until now — are types of resources. It therefore adds the meta-facts "type-of(gasoline, resource)" and "type-of(oil, resource)" and the conclusions "can-apply(requires, engine, gasoline)" and "can-apply(requires, engine, oil)" to its meta-fact base for future reference. (Note that it only draws "can_apply" conclusions from specific factual instances; the conjecture processing subsystem, however, which we do not discuss in this paper, adds the "must_apply" versions of these versions to the conjecture stack to await possible disconfirmation by later instances mentioned in the dialogue.) Thus, partial knowledge about two new words has been acquired just from the

context provided by the question. Now suppose that the rule "must-apply(produces, process, resource)" is in the ontology. From this meta-knowledge and the newly acquired meta-fact "type-of(gasoline, resource)", KRIMB deduces that there must exist some process that produces the gasoline flow consumed by engine (01). Hence, it adds the question: "WHAT PROCESS PRODUCES GASOLINE FLOW(02)?" to its stack of pending questions. A similar question is generated for oil. The knowledge from the last question-answer exchange has now been processed, and the next question is selected from the pending question stack and asked to the user.

### D. Structuring the Interview

The knowledge-acquisition process described above uses a default control strategy based on model completion to guide the order of questioning. A set of nodes (each representing failure of a specific subsystem or "process") within the model is said to be logically closed if (i) all applicable relations for the nodes have been instantiated, i.e., the input and output flows and components (if any) of each node have been recorded in the fact base; and (ii) the resulting network of relations on nodes is connected, i.e., there is a chain of relations between any two nodes in the set. In our example above, the only relation between nodes was or-enclosure; more generally, other logical and temporal precedence constraints can be used as well. A logically closed set of nodes is called a module. For our example, modules represent failures of systems or subsystems. KRIMB generates questions by seeking to instantiate relations that will close whatever module it is working on. Breadth-first expansion is used to initially identify the subsystems of a system, and input flows are traced to identify other parts of the system. If a system has no identified subsystems, then it must be assigned a failure probability.

It is worthwhile to ask what kind of knowledge is acquired through this kind of process. The knowledge represented in a reliability model is not primarily knowledge of facts, but knowledge of possibilities. Of course, facts constrain possibilities, which is why questions about facts are asked during the model-building phase. But a reliability model is used to answer questions like "What is the probability that subsystem X will fail within a year?" or "If subsystem X fails after a year, what is the probability that failure of component C will be one of the causes of failure?" To reason out the answers to such questions, KRIMB does not have to consider the passage of physical time, as in a simulation model. Only the possible sequences of events that could lead to the specified outcome, as determined by the precedence constraint partial ordering, are important. Thus, KRIMB exploits a very weak form of temporal reasoning, specifying that one event becomes possible only when other events (its preconditions) are completed. This type of reasoning is sufficient not only for building reliability models, but for building causal models, decision trees, influence diagrams, and other forward-looking structures that are concerned primarily with inferring knowledge about possibilities from knowledge about facts.

## 4. Conclusions

This paper has described a knowledge acquisition system that uses goal-directed, intelligent questioning, drawing on relevant meta-knowledge about the application domain (e.g., air conditioners, nuclear power plants, etc.) to acquire knowledge about a specific system. A logical model of the system is constructed and displayed to the user during the knowledge acquisition dialogue. Users may make assertions, but the system will resume questioning under its own

initiative until it detects that the model it has built is "closed" under all applicable inferences. The model is then used to calculate the reliability of the system.

KRIMB demonstrates several innovative approaches to knowledge acquisition. Perhaps the most important of these is the use of an ontology to pre-structure the kinds of questions that get asked (instantiation and specialization questions) and the kinds of facts and meta-facts that can be acquired and represented. Only knowledge that will be useful in model-building is sought, and the process of model-building sets the agenda of the acquisition interview. Finally, the idea that only knowledge that can be *represented* (through facts and meta-facts) and *used* (to build models) should be acquired provides a source of conceptual economy in generating questions and helps integrate the knowledge representation and knowledge acquisition sides of the knowledge engineering enterprise.

**ACKNOWLEDGEMENT**

The authors gratefully acknowledge the thoughtful review and suggestions of Paul Bauer of U S WEST and Jonathan Bein of the University of Colorado.

**References**


[Al 86] Alexander, J., Freiling, M., Shulman, S., Staley, J., Rehfuss, S., and Messick, S., "Knowledge Level Engineering: Ontological Analysis," *Proceedings of the National Conference on Artificial Intelligence.* Philadelphia, PA 1986.

[Bo 85] Boose, J., "Personal Construct Theory and the Transfer of Human Expertise," *Proceedings of the National Conference on Artificial Intelligence.* Austin, Texas 1984.

[Br 85] Brachman, R. J., and Schmolze, J. G., "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science, 9, 2, April, 1985.

[Ca 83] Card, S., Moran, T., and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Earlbaum Associates, 1983.

[Co 87] Cox, L. A., and Blumenthal, R., "Dynamic Planning Under Uncertainty Using Automated Model Construction and Risk Analysis," *Proceedings of the 1987 Workshop on Simulation and Artificial Intelligence.* Seattle, WA 1987.

[Co 86] Cox, L.A., "A Probabilistic Risk Analysis Program for Analyzing Security Risks," *Proceedings of the 1986 Society of Risk Analysis Annual Meeting,* Plenum Press, 1987 (forthcoming.)

[Ha 83] Haas, N., and Hendrix, G., "Learning by Being Told: Acquiring Knowledge for Information Management," In Machine Learning, Mitchell (ed.), Tioga Publishing, 1983.

[Gr 85] Grosz, B., and Sidner, C. "Discourse Structure and the Proper Treatment of Interruptions," *Proceedings of the Ninth International Conference on Artificial Intelligence.* Los Angeles, CA 1985.

[La 86] Laird, P.D., "Inductive Inference by Refinement," *Proceedings of the National Conference on Artificial Intelligence.* Philadelphia, PA 1986.

[Mi 83] Mitchell, T., Utgoff, P., and Banerji, R., "Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics," In *Machine Learning.* Tioga 1983.

[Qu 80] Quine, W., From a Logical Point of View, Harvard University Press, Cambridge, MA 1980.

[Qu 86] Quinlan, J.R., "The Effect of Noise on Concept Learning," in Michalski, R.S. et al, *Machine Learning: An Artificial Intelligence Approach, Vol. 2*, Morgan Kaufman. Los Altos, CA 1986.

[Sa 86] Sammut, C. and Banerji, R.B., "Learning Concepts by Asking Questions," in Michalski, R.S. et al, *Machine Learning: An Artificial Intelligence Approach,* Morgan Kaufman, Los Altos, CA 1987.

[Sc 82] Schmolze, J. G., and Brachman, R. J., *Proceedings of the 1981 KL-ONE Workshop*, Bolt, Beranek, and Newman, Technical Report 4542, 1982.

[Sh 85] Shastri, L., and Feldman, J., "Evidential Reasoning in Semantic Networks: A Formal Theory," *Proceedings of the Ninth International Conference on Artificial Intelligence.* Los Angeles, CA 1985.

[Si 83] Sidner, C., Bates, M., Bobrow, R. Goodman, B., Haas, A., Ingria, R. Israel, D., MaAllester, D., Moser, M., Schmolze, J., and Vilan, M., Bolt Beranek and Newman Report 5421, 1983.

[So 84] Sowa, J. *Conceptual Structures Information Processing in Mind and Machine,* Addison-Wesley Publishing, 1984.

[Sl 83] Slagle, J. and Gaynor, J., "Expert System Consultation Control Strategy," *Proceedings of the National Conference on Artificial Intelligence.* Washingtion, D.C. 1983.

[Vi 84] Vilan, M., "KL-TWO, A Hybrid Knowledge Representation System," Bolt, Beranek, and Newman, Technical Report 5694, 1984.

# A METHODOLOGY FOR INTEGRATING KNOWLEDGE ACQUISITION TECHNIQUES

Joachim Diederich*
Marc Linster*
Ingo Ruhmann*
Thomas Uthmann+

*Institute for Applied Information Technology
+Institute for Technology Transfer
German Research Institute for Mathematics and Data Processing
Schloss Birlinghoven
Postfach 1240
D-5205 Sankt Augustin 1
West Germany

**Abstract**

We present a general purpose knowledge acquisition tool that uses the advantages of domain specific tools. To adapt the knowledge acquisition tool for a specific application and a specific problem solving strategy, acquisition knowledge bases (AKBs) are integrated in the system to guide the employment of different knowledge elicitation methods (interview techniques, protocol analysis, semantic text analysis and learning mechanisms). Acquisition knowledge bases are predefined deep models, consisting of structured objects to represent important concepts of a domain. We propose an architecture to make exhaustive use of hybrid knowledge acquisition modules. An augmented interview will be introduced to comply with the demands for higher efficiency. Furthermore, methods for integrating these knowledge acquisition tools with machine learning approaches are discussed.

An example introduces some important features of the methods used in KRITON.

## A. Introduction

The knowledge acquisition tool KRITON is designed for use by inexperienced users, unfamiliar with AI techniques (e.g. a domain expert) and by well-trained knowledge engineers. KRITON is a general purpose knowledge acquisition tool incorporating different elicitation methods, such as interview techniques, protocol analysis, semantic text analysis and machine learning methods.

Therefore KRITON is a hybrid knowledge acquisition tool comparable to AQUINAS (Boose and Bradshaw 1987) and KADS (Hayward, Wielinga and Breuker 1987).

Each knowledge elicitation method is used for capturing knowledge from only one specific knowledge source: 1) interview methods, using the repertory-grid techniques also used in (Boose and Bradshaw 1987) and a laddering component for an analysis of the taxonomic structure of a domain, are employed for the acquisition of declarative knowledge; 2) protocol analysis, related to the original work in (Waterman and

Newell 1971), is restricted to the elicitation of procedural knowledge; and 3) semantic text analysis is used to search for important text fragments, frequency analysis of relevant keywords and the generation of semantic text structures. For a more complete description of the use of these methods in KRITON see (Diederich, Ruhmann and May 1987, Diederich 1987-b).

All output of the above mentioned elicitation methods is represented in an Intermediate Knowledge Representation Level. The purpose of this intermediate knowledge representation is the storage of the already available knowledge in a way that makes it possible to employ different knowledge elicitation methods.

The aim of this paper is an introduction to the method of knowledge-based knowledge elicitation through several elicitation methods in a hybrid knowledge acquisition tool. This is done by using intermediate stages of knowledge representation as a blackboard for further refinement.

## B. Utilization of Already Acquired Knowledge

The employment and use of the already acquired knowledge has major advantages and is an important task for knowledge acquisition tools. General purpose knowledge acquisition systems like KRITON are not so complete in their support of the knowledge acquisition process as domain-specific systems. For example, OPAL (Musen 1987) is complete for the domain of cancer therapy. The only chance for a general purpose knowledge acquisition tool to achieve such a capability is to use as much as possible of the already acquired knowledge to adapt the system for a specific domain.

Another problem arises if different problem solving strategies (e.g. heuristic classification) are to be supported. In this case meta-knowledge is necessary to realize different strategies of applying the available knowledge.

In KRITON, already captured knowledge is used in several ways, depending on its amount and quality. Moreover, existing knowledge is completed by "acquisition knowledge bases" (AKBs) for better guidance of the ongoing elicitation process. These acquisition knowledge bases are viewed as an integral part of the KRITON system. In every stage of the acquisition process, the user can use these knowledge bases in addition to existing knowledge for better employment of the KRITON facilities for knowledge-based knowledge elicitation.

AKBs contain a set of structured objects defining important concepts of the domain. They are predefined declarative deep models of a domain with the sole purpose of optimizing the ongoing acquisition process. There is no guarantee that the AKBs themselves are complete or consistent: they are a result of an acquisition process and they are used only for the purpose of further knowledge elicitation.

Depending on the richness and quality of the existing knowledge, the already acquired knowledge is used in the following ways:

* Subject of further specialization processes.

* Guidance of the acquisition process by discovery of missing components.

* Completion of domain-dependent deep models (AKBs).

Section D will give an introduction to the use of already captured knowledge in addition to AKBs.

### C. Intermediate Knowledge Representation Level

Some knowledge acquisition tools include different knowledge representation stages to represent intermediate results of the total knowledge acquisition process, e.g. OPAL (Musen 1987) and KADS (Hayward et al. 1987). Especially if learning mechanisms come into play, an intermediate knowledge representation level is useful to maintain information closer to the sources. The step from generated rules back to the original facts is often necessary for evaluating the results of the acquisition process.

In our system, all output from the knowledge elicitation techniques is translated into an intermediate knowledge representation system. This representation system has two subparts: a descriptive language for functional and physical objects, representing the generic concepts, and a propositional calculus representing the transformation path of these concepts during the problem solving process.

The description language consists of structured objects, their features and interrelations and is used to define semantic relations in a domain. The classification of taxonomic relations is similar to that in (Brachman 1983). This semantic net is the goal language for the interview and textual analysis methods.

The second part of the intermediate knowledge representation language is a set of operator-argument structures using semantic primitives to describe the basic relations of concepts discovered by protocol analysis. The set of semantic primitives is not complete and will have to be updated for each application domain (e.g. engineering).

The intermediate knowledge representation level allows the use different knowledge sources and permits the tool to be extended with elicitation methods currently not available. Moreover, it can be used for the generation of various knowledge bases for different expert system shells and knowledge representation systems.

From this point of view, an Intermediate Knowledge Representation supports at least these six desirable properties:

* Openness of the system (possible integration of currently unknown elicitation methods).

* Knowledge acquisition for different knowledge representation languages.

* Storage of incomplete and inconsistent knowledge for the ongoing acquisition process.

* Integration and employment of acquisition knowledge bases.

* Maintaining information closer to the sources (e.g. through references to the original expert utterances, protocols etc.).

* Management of knowledge bases with varying degrees of completeness in different knowledge representation languages.

## D. Using Incomplete Knowledge for Refinement and Specialization

The discovery of incomplete knowledge and the search for hypotheses with best evidence is one important feature of knowledge acquisition tools, e.g. MOLE (Eshelman, Ehret, McDermott and Tan 1987).

In KRITON, the employment of a certain knowledge elicitation method depends not only on decisions of the knowledge engineer but also on requirements the system discovers by analysis of the already acquired knowledge.

A significant role in dealing with incomplete knowledge is played by "Watcher", which is a permanently active demon controlling the intermediate knowledge representation for missing components.

The user (the knowledge engineer or expert) might have generated several objects during the incremental text analysis without any relation to the taxonomic organization of the objects of the corresponding domain (e.g. no information about the inheritance paths, part-of relations or instance relations was given). These items are known to the systems, but should be subject of further investigations. Watcher checks all objects at the intermediate knowledge representation level for missing, but possible or indispensable links (e.g. every object has to be placed in a taxonomic organization), sends a message to the user and recommends the employment of an elicitation method to complete the knowledge base.

Here is a simple example. As part of an AKB, the system has stored at the intermediate knowledge representation level a "part-of" relation between the objects "motor" and "car". Watcher will discover this relation and will trigger an interview. More precisely, Watcher selects a question from the laddering component to explore the "part-of" relation between "motor" and "car". First, "physical-part-of" relations are analyzed, next "subset" or "generalization" relations and so on, until the most specialized relation is found.

Watcher is also invoked if an elicitation method starts, informing the user about incomplete parts of the knowledge base. Furthermore, the user can delegate the selection of concepts to be used in an interview to Watcher. This demon then looks for semantically related but incomplete objects and triggers a laddering to exploring that domain in more detail.

To deal with incomplete knowledge and use this kind of knowledge for specialization tasks, Watcher clearly needs meta-knowledge.

The discovery of missing objects and links in a semantic structure is only possible by using knowledge about the required semantic relations for an object. At this point,

knowledge about semantic structuring in KRITON comes into play. Although Watcher is realized purely procedurally now, the KRITON approach could profit from a knowledge-based organization of Watcher.

The general aim of this approach is to extend the facilities of the knowledge based Watcher. The purpose of this component is not only to recommend knowledge elicitation methods but also to automatically employ them.

## E. Supporting Problem-Specific Knowledge Acquisition in a Hybrid Architecture

In the first paragraphs of this paper we have discussed how a variety of methods can be used in different domains. Now we shall look at the possibilities to use them for different problem solving strategies.

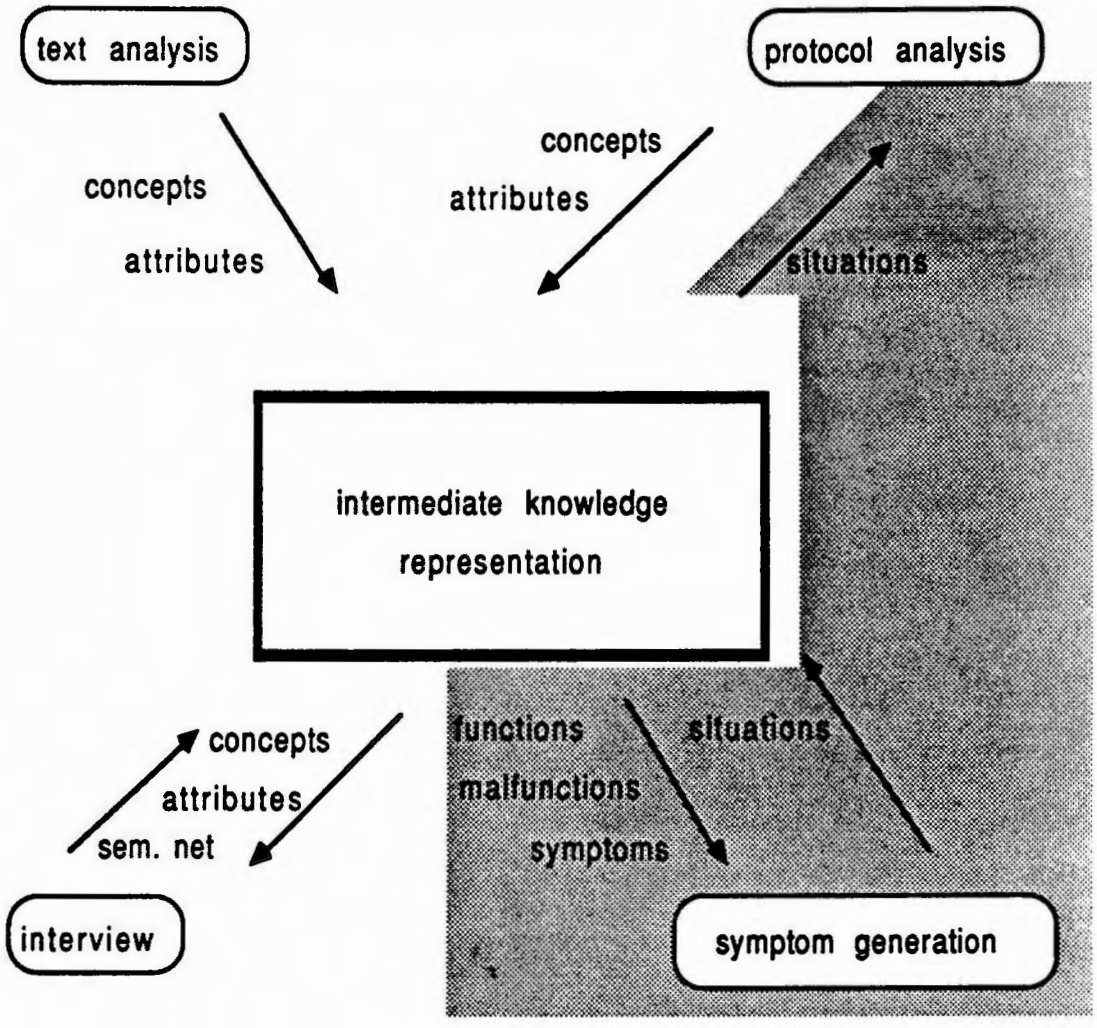
No single knowledge acquisition method seems strong enough to acquire knowledge for all discourse domains and all problem solving strategies. Interview techniques, like repertory-grid or laddering, are fit for the acquisition of knowledge for classification and differential diagnosis. Protocol analysis acquires knowledge for solely rule based systems, text analysis is still of very limited use as a stand-alone-procedure.

KRITON proposes a hybrid acquisition method based on text analysis, interview techniques and protocol analysis. KRITON is not just an assembly of these three loosely connected methods, but it is strongly based upon using the advantages of each technique in the domain that the technique is best at. The symbioses of the three techniques is organized like a blackboard model.

Text analysis provides the intermediate knowledge representation (IKR) with concepts and attributes. The interview uses what is present in the IKR to build, complete or modify a semantic network in a dialogue with the expert. The semantic network contains, in the case knowledge for diagnosis is acquired, a subgraph of functions, malfunctions and symptoms. Their combinations can be used to construct relevant situations to trigger a protocol analysis. The protocol analysis feeds concepts and attributes back to the IKR as they occur during the thinking aloud process of the expert. This interaction allows for an incremental process of knowledge acquisition, each method pitching in what it is best at. As there are no conflicts between methods, because there ranges are distinct, we have no need for conflict resolution on this epistemological level.

AKB´s can be introduced into the IKR as generic knowledge bases or as skeletons of knowledge bases. If the AKB is the sketch of a taxonomy, the acquisition procedures are directed toward the elicitation of static knowledge, whereas if it is a set of triggering cases, the acquisition of procedural knowledge will be stressed.

Organization of the acquisition techniques and their interaction




The shaded area includes a method specific to diagnostic problem solving.

## F. An Augmented Interview to Increase Efficiency

In the interview the static non-procedural aspects of the domain are elicited. The elicitation consists of a differentiation between concepts and a structuring of concepts. Neutrality is a very important property that interview techniques should have. Classic repertory-grid techniques show good performance if they are applied on concepts belonging to the same level of abstraction (Boose and Bradshaw 1987), but they are of little use if hierarchical structures are encountered. Laddering techniques (Diederich 1987) offer to impose structure upon concepts, but these techniques do still have the same complexity problem that repertory-grid has. We propose a mechanism that lets the user build a taxonomic structure upon a set of concepts in a first step. In a second

step levels of abstraction in the user-defined structure are analyzed with a repertory-grid technique. In a third step a classifier tries to rebuild the original taxonomic structure by asking more questions about concepts, attributes and their values. This method seems to guarantee neutrality towards the expert, as he does not have to build a correct structure, it suffices if he makes a rough sketch, that may be totally wrong. This will not have any effect upon the resulting network, it will only lead to a sensibly larger set of questions asked by the system.

An important feature of this interview is the representation of constructs as attributes and their values. This seems to be in conflict with George Kelly´s personal construct theory (Sader 1980), especially with the dichotomy-corollary "The human construct system consists of a limited amount of dichotomic constructs" (own translation). We argue that "color(car) = green" and "not(color(car) = green)" are dichotomic constructs too. As Kelly allowed constructs like "low morals" or "less understanding" (Sader 1980) in his repertory-grid-tests, and as other knowledge acquisition systems allow the expert to rate constructs to express how much a construct applies to a concept (Boose and Bradshaw 1987), we think that the concept of dichotomic constructs may be applied to attributes and values, without leaving the theoretical foundations of personal construct theory.
This augmented interview is not yet part of KRITON, but will be integrated in the next version.

## G. Integrating Machine Learning and Psychological Elicitation Methods

There are several ways to integrate machine learning with cognitive science methods. First, learning mechanisms can be used as knowledge elicitation methods, together with interview techniques, incremental textual analysis and protocol analysis. Second, learning mechanisms can operate on structured objects at the intermediate knowledge representation level to find similarities between them and recommend a certain organization of the structured objects (e.g. if an analogy is discovered). Third, the learning component can observe user activities on line, support the user with feedback about regularities in his use of the system and display connected items in the form of rules. This last possibility has been adopted in KRITON.

The KRITON architecture supports such an integration. The intermediate knowledge representation system allows the storage of knowledge together with references to the sources of a particular entry. This facility avoids a major disadvantage of learning systems. In KRITON, the step back from generated rules to the original facts and sources is always possible by special markers at the intermediate knowledge representation level.

In addition, the above described application of learning mechanisms supports the knowledge-based knowledge elicitation process. Because the state of the existing knowledge in the system may always change, additional information about similarities and regularities of parts of the intermediate knowledge representation is necessary.

COLE (COncept LEarner) is a learning-by-example system similar to ACLS (Paterson and Niblett 1982). It uses induction mechanisms. The induction-algorithm in COLE is a modification of Quinlans ID3 (Quinlan 1979). From an input of classes and their attributes COLE induces concepts of these classes. These concepts allow conclusions from attributes to classes.

COLE does not, like other learning systems generate a decision tree, but it produces rules in the form of Horn-clauses. The reason for that is the lack of clarity in decision-trees when they reach a certain size and complexity. Rules in the form of Horn-clauses are not only much more compact, but are the representation-form used in many expert-systems. In KRITON, COLE translates the Horn-clauses into BABYLON-Format, but rules can be generated for any other expert-system-shell as well.

The presently implemented version concentrates on the use of COLE as an acquisition method. Just like the other methods in KRITON, COLE allows the user to make the input in a non-formal, simple, and natural language-like syntax. Attributes, their values and the classes they characterize are analyzed through a keyword-recognition mechanism.

When the user has finished the input, COLE checks all classes whether they were assigned an identical set of attributes and values. If a duplicate is found, COLE asks the user if the reason is due to

- incorrect input,
- an undetected hierarchical relation,
- the lack of a discriminatory attribute.

This procedure is repeated until every class of objects is unequivocally defined through a set of attributes and the corresponding values. The rule generation tries to build the simplest rule, i.e. the rule with the shortest if-part. It tries to find unequivocal attribute values for each class of objects. Ranges of values are represented with a disjunction.

Conceptual learning in COLE is to be used as a starting point for the employment of other learning techniques in KRITON. The next step will be the conversion of the input acquired by COLE and by the other acquisition-methods in KRITON into a formal concept lattice as used and developed in (Wille 1982). Its advantages are the completeness as well as the clearness of representation and the reduction of computation time, especially with larger data-sets. A formal concept lattice allows also the integration of methods from similarity-based-learning.

## H. Examples

A sample knowledge base for motordiagnosis will be partly generated by use of several knowledge acquisition methods.

The text analysis of a technical description generates some objects:

engine, carburator, cylinder, piston, crankshaft etc.

As these objects are not yet integrated into the knowledge base, the watcher triggers an interview. The interview component, as part of the repertory grid technique presents them to the expert as triplets.

INTERVIEW

*KRITON : Which property do carburator and cylinder share, that piston does not have?*
*Expert : They are containers.*

*KRITON : Which property do cylinder and engine share, that piston does not have?*
*Expert : None.*

Here KRITON switches from the repertory grid technique to a laddering technique to elicit taxonomic structures.

*KRITON : Is piston a part of cylinder and engine?*
*Expert : Yes.*
etc.

The result of the repertory grid and laddering techniques is represented as declarative knowledge in a frame-like syntax in the intermediate knowledge representation language.

PROTOCOL ANALYSIS

The protocol analysis, employed for the acquisition of procedural knowledge, is used independently of the interview technique. The transcription of the experts thinking aloud protocol contains markers indicating pauses within the experts speech.

*Expert: If the oil is burned and brown ** the clutch and the breakbelts are damaged.*

This sentence is segmented :

(D1 If the oil is burned and brown)
(D2 the clutch and the breakbelts are damaged)

Further transformation produces :

(burned D1 oil)
(brown D1 oil)
(damaged D2 clutch breakbelts)

This propositional representation uses domain-dependent primitives which have to be elicited with the interview techniques to assure that the rules KRITON generates are backed up with declarative knowledge.

CONCEPT LEARNING

The concept learner COLE defines classes through the values they have for sets of attributes.

*KRITON : Give the value for the class oil and the attribute color.*
*Expert : Yellow.*
*KRITON : Give the value for the class oil and the attribute viscosity.*
*Expert : Low.*
*KRITON : Give the value for the class breakfluid and the attribute color.*
*Expert : Yellow.*
*KRITON : Give the value for the class breakfluid and the attribute viscosity.*
*Expert : High.*

COLE allows the generation of rules like :

IF color yellow
viscosity high
subclass-of liquids
THEN class is breakfluid.

## G. Conclusion

We are aiming at an integrated, modular tool for knowledge acquisition in knowledge-based systems. The system should be open in the sense that it provides facilities for its own extension and elaboration.

Important extensions, which have already been implemented, include acquisition knowledge bases for the adaptation of the system to a certain domain and to a a certain problem solving strategy. Acquisition knowledge bases are used for the employment of knowledge elicitation methods to specialize and complete the already acquired knowledge.

## REFERENCES

Boose, J.H. and J.M. Bradshaw "Expertise Transfer and Complex Problems Using AQUINAS as a Knowledge Acquisition Workbench for ExpertSystems." In: Boose, J.H. and Gaines, B. (Eds.): Proceedings of the "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada (toappear at Academic Press 1987).

Brachman, R.J. "What IS-A is and isn't: an analysis of taxonomic links in semantic networks." IEEE Computer, 16 (special issue: Knowledge Representation), 1983

Diederich, J., I. Ruhmann and M. May "KRITON: A Knowledge Acquisition Tool For Expert Systems." In: Boose, J.H. and Gaines, B. (Eds.): Proceedings of the "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada (to appear at Academic Press 1987).

Diederich,J. "Knowledge Based Knowledge Elicitation." In: Proceedings of "IJCAI 1987", Milano, Italy

Diederich, J. "Wissensakquisition" GMD Arbeitsbericht 245, St. Augustin, 1987
Eshelman, L., D. Ehret, J. McDermott and M. Tan "MOLE: A Tenacious Knowledge Acquisition Tool." In: Boose, J.H. and Gaines, B. (Eds.): Proceedings of the "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada (to appear at Academic Press 1987).

Hayward, S.A., B.J. Wielinga and J.A. Breuker "Structured Analysis of Knowledge." In: Boose, J.H. and Gaines, B. (Eds.): Proceedings of the "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada (to appear at Academic Press 1987).

Musen, M.A. "Using a Domain Model to Drive an Interactive Knowledge Editing Tool." In: Boose, J.H. and Gaines, B. (Eds.): Proceedings of the "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada (to appear at Academic Press 1987).

Waterman, D.A. and A. Newell "Protocol Analysis as a Task for Artificial Intelligence." Artificial Intelligence, 2 (1971), 285-318.

Paterson, A. and T. Niblett, "ACLS User Manual (Apple-ACLS Version)" ITL, 1982,2 Hope ParkSquare, Edinburgh EH8 9NW, Scotland

Quinlan, J. R. "Discovering Rules by Induction from Large Collection of Examples" In: Expert Systems in the Micro-Electronic Age (ed.: D. Michie), Edinburgh University Press, Edinburgh, 1979, pp. 168-201

Wille, R. "Liniendiagramme hierarchischer Begriffssysteme".Preprint Nr. 812, TH Darmstadt, FB Mathematik, April 1984

Sader M. "Psychologie der Persönlichkeit" Juventa Verlag, München 1980

# KET, A Knowledge Encoding Tool

*Ladan Esfahani and Dr. F.N. Teskey*

Brighton Polytechnic, IT Research Institute, Brighton BN2 4GJ.

*ABSTRACT*

This paper describes KET ,a knowledge encoding tool, which is a contribution towards knowledge elicitation in expert systems. KET's aim is to give the domain expert as much freedom as possible. KET presents an intelligent interface that creates a simple user's model for experts with some computer experience. KET is general enough to encompass a wide range of expert systems; whether they are of interpretation or consultation type or whether their fields are as diverse as project planning or archaeology. One of KET's tasks is to guide the expert with rules and their consistency which could be a step towards second generation expert systems. Further development will include incorporation of rule learning ability in order to narrow the rule inquiry process from the expert.

KET is one of the tools in our AI research environment, IDEAS. IDEAS is built around a knowledge representation language, RBFS that uses a combination of frame-based and rule-based paradigms. Simplistically, RBFS consists of a semantic network tool describing information structures, augmented by production rules describing expert's knowledge. IDEAS also includes additional tools such as a visual editor, VEGAN. VEGAN basically enables creation of semantic networks, and therefore includes a graphical editor with the necessary viewing operators. Experts should be able to encode their rules with KET in conjunction with Vegan. KET is then mapped into RBFS, for its underlying code. VEGAN was developed to visualise the frame-based aspect of RBFS and KET complements this by visualising the rule-based side of it.

KET is implemented in POP-11 language on the SUN workstation.

## 1. Introduction

Knowledge elicitation is the most elusive of all Expert Systems skills, the most difficult to acquire and the subject about which least written; hence, its reputation as the bottleneck of Expert Systems development.[1] Interviewing the experts has been the most successful and complete methods of Knowledge Elicitation so far. Automation of knowledge elicitation has been suggested in the past but a truly successful tool has not yet been developed. The current automation methods have been more of an aid to the Expert rather than a replacement of the Knowledge Engineer.[2] Methods such as induction have been implemented but as yet have their deficiencies. Induction is really effective when used in conjunction with interviewing techniques. See Hart[3] for problems of this method.

Automation has posed such a problem that as knowledge engineers, we have our reservations as to whether this is a feasible task at all. It may be possible that elicitation will be completely automated in future, but this does not seem to be feasible for the time being. Hence, we have taken the prudent view here of not claiming to have produced a tool for automation of knowledge elicitation, but have created a simple user's model and guidance to ease experts' task in encoding their knowledge into the system.

In this paper we first outline the underlying history to such a tool, followed by its description, its structure and functions, and finally how we propose to complete it as a more intelligent tool in future.

This we hope to achieve by delving into the field of machine learning. Our guidance will be our researchers' experience and needs in the field of expert systems. Please note that the term 'user' in this paper applies to the expert who assists in building the system and not the client, unless stated otherwise.

## 2. Background :

### 2.1. RBFS, The Knowledge Representation Language :

RBFS (Rule Based Frame System) is the underlying knowledge representation language for KET. Here, we will explain its functions where it concerns KET but for a more detailed description please consult the relevant literatures[4] (Also see Eklund 1987).[5]

RBFS is a knowledge representation language that was developed in our research environment tailored to the need of the Expert Systems programmers. The original Expert Systems using RBFS were written for application areas such as project management, bid proposals and functional test strategies for printed circuits. RBFS's functions were inspired after studying EMYCIN, hence a few similarities exist between RBFS predicate functions and EMYCIN's.

#### 2.1.1. Epistemological Issues :

Frame-Based representation constitutes the kernel underlying the conceptual model of RBFS. Also experts reason using fundamental principles of their domain, which provide additional justification, especially in those cases not covered by the rules. This was realised by the developers of RBFS who consequently incorporated this need in the form of Semantic Networks. This network representation was implemented to illustrate the Frame-Based aspect of RBFS. This is one form of representing deep reasoning,[6] It also allows implementation of inheritance property as is one of the characteristics adhered to Associative Networks.

RBFS uses an additional representation, namely rule-based representation. The reason is that experts tend to utilise expertise, translated into 'rules of thumb' as well as shortcuts that tend to lead quickly to the solution and eliminate inflexible alternatives for uncommon problems. This is generally known as surface reasoning.

The use of this combination of representations is emphasised in several literatures[7] (also see GEOTOX 1987). [6] and therefore was a reinforcement of RBFS developers' convictions. They advanced their system by incorporating a graphical network for demonstrating their Expert Systems and also to aid in eliciting knowledge from the domain experts such as managers and engineers. Semantic networks seemed to be an easy visual aid and easily grasped representation for the experts of the specified fields. The use of natural languages was dismissed for the significant difficulties that have been encountered in parsing natural languages [8] (See Davis 1979). [9] Also many psychologists feel strongly that verbal data are useless [10] (See Bainbridge). [11]

Originally, therefore, the graphical implementation of semantic networks was incorporated as part of the Expert Systems written in RBFS. Eventually came a recognition that this graphical network needed an independent identity and could act as a separate interface to expert systems. This would depict clearly the general software subsystems, and would aid visualisation of experts' interaction within a computer integrated system. This is how VEGAN was born.

### 2.2. VEGAN the Visual Editor :

VEGAN (Visual Editor for the Generation of Associative Networks) was developed by a Research Assistant in our department.[12] VEGAN basically caters for visualisation of the Frame-Based aspect of RBFS and relies on Semantic Networks for this purpose.

#### 2.2.1. Epistemological Assumptions :

The major concepts which exist in VEGAN can be itemised as planes, nodes and links. The set of associations and concepts that make up a definition is called a "plane". [13] Nodes generally represent objects and links indicate the relationship among those objects in any given field. Nodes are further divided into two categories. In the first category lie nodes which are generic and descriptive of other

nodes. These are concepts being defined and are called "type nodes". Their definitions are determined upon other generic concepts to which they link. The plane where they reside is called the general plane. The concept of general planes has been considered as one way of Meta-knowledge implementation. [14]

The second category of nodes consists of those which are particular instances of the first category. These are called "token nodes". Token nodes reside in the specific plane and are connected to their type nodes via 'inst' links in VEGAN terminology.

A distinction is made among the links, too. Links are categorised into 'declarative' type and 'inference' type but are syntactically named differently. The 'declarative' type indicates that the relationship shown on the network exists and the values are assigned to the appropriate frames. The evaluation of the frames connected via the 'inference' links are postponed till the inference of the rules. 'Inference' links are what KET is concerned with. Nodes in VEGAN represent the frames and links represent the slots in the frame-based terminology. For a more detailed description of VEGAN see Kellett. [12]

The semantic network of object and relationships could be created via VEGAN and the rules which conclude conditional relationships, were written in RBFS. This hybrid scheme between RBFS and VEGAN is implemented in POP-11, on the Sun workstations. A knowledge of POP-11 programming was essential when programming in RBFS. Therefore, the need for visualising the rule base aspect of RBFS remained unfulfilled.

### 2.3. Birth of KET :

KET was needed in order to facilitate rule writing in RBFS. KET relieves the domain experts or the knowledge engineers from building systems in an artificial intelligence language by providing an easily used graphical interface.

KET not only caters for visualisation of the rule aspect of RBFS but also provides a way for the expert to think about how rules and their contents are related. For example, If one wishes to conclude about the truth of a certain concept or relationship eg. buying a house, one will have to write a set of appropriate rules. However, conclusion of these rules will be dependent on the existence of some further concepts and relationships, and the rules that conclude those relationships and so on. Therefore, an inverse semi-tree structure will result with rules and concepts related as nodes or leaves of the tree. At the time when programmers were relying on RBFS for building Expert Systems, they systematically made a dry run of this structure to find the inter-dependence of relationships and the relevant rules. This was the 'user's model' that the users manually constructed to illustrate their programs to themselves. Therefore it was only natural that KET would use this model to satisfy the users' need in building the rules.

### 2.4. Visualising RBFS Functions :

Before explaining KET's functions in more detail, we'll have to explain the terms used in RBFS with a closer scrutiny. We will restrict ourselves to those functions that the Expert would be manipulating. For a more detailed description of RBFS Functions please refer to Marshal 86. [4]

An RBFS program could be envisaged as having three modules, namely, Conclude, Procedure and Ask; briefly, these are the three ways in which the system could derive values for its slots. In the RBFS language they refer to inference by rules, calculating via procedures and asking the values from the client directly; respectively. They form an independent concept called facets in RBFS. Additional facets such as direct evaluation of the frame-slot relationships or assignment of default values exist but VEGAN has already catered for these evaluations and consequently their discussion is omited in this paper. Facets could be thought of as another way of implementing Meta-knowledge. [14]

#### 2.4.1. CONCLUDE :

First part of any RBFS program would CONCLUDE about the truth or existence of slots or relationships. This is the heart of the inference mechanism of RBFS. This module consists of a set of rules which embed a set of premises each of which run a selection of predicates. The predicates are basically variations on checking the existence or equality of different frames. ie. they are different

kinds of conditions in an "IF" statement. Examples of predicates are : Less, Same.

Attached to the truth or otherwise of each of these rules is a certainty factor (cf), which the expert recognises as the strength of that premise in concluding the truth of a specific value for a slot. RBFS uses the notion of certainty factors not to be confused with probabilities. They are based on EMYCIN. A fragment of a generalised rule module is shown below.

```
CONCLUDE frame slot;

defrule ruleno.;

  [ [list of predicates]
  => cf value]
endrule;

endCONCLUDE;
```

NB. A set of predicates define a premise which "conclude"s a concept.

**2.4.2. Procedures :**

The flexibility of RBFS comes from its ability to include procedures that might be specific to the application field. These procedures are written in POP11. The visual interfaces (VEGAN and KET) do not cater for writing procedures, for this will require an interface to the POP11 language. A set of library procedures for some applications will be incorporated in the near future. These procedures would be accessible to the user via an appropriate menu option.

**2.4.3. Ask :**

The last place that instantiation of a slot may lie is in the ask section. This section contains the questions asked from the client. The expert should also specify the range of accepted answers. This can be illustrated as follows:

```
defask frame slot;

  'allowable values' -> range;
  'question text' -> question;

endask;
```

NB. "range" and "question" are predefined variables.

**3. KET's Functions :**

First step is generation of the 'information base' via VEGAN. The uncertainties are then dealt with via KET. KET has kept VEGAN's notations to some extent in order to safeguard the consistency between the two tools. However, where they have conceptual differences they also differ in their representation. Concepts which would be different from nodes and links in KET are classified as rules, predicates, certainty factors and facets. Like VEGAN, interaction with KET is on a WIMP basis.

KET, therefore, resembles the conceptual graph representation more closely than associative networks. For supportive arguments for conceptual graph representation see Sowa. [15]

VEGAN and KET are able to drive one another graphically and therefore, allow their users to alternate between the two. When an operation is acted upon either one of them which might have an effect on the graphical structure of the other, the first calls a similar function which would modify the second accordingly. This would mean that for example when a node representing a frame is deleted in the Vegan graph, Vegan in turn calls a KET function that would delete the same frame and its associations in the KET graph.

As we said, the semi-tree structure forms the basic representation of KET. After entering KET the expert starts by specifying the object and the relationship for which a value is to be derived. This is done in the form of nodes and links as described before.

As an example we will take a hypothetical case where the system would advice a client in buying a house. We address this problem for its comprehensibility by all disciplines.

In this case the expert would be writing rules to CONCLUDE whether a "person" (the frame) "should_buy" (the slot) a "house" (the value-frame). A possible program in RBFS is shown in figure 1. A corresponding representation in KET is illustrated in figure 3. Certainty factor is represented along side the rulename. Also note the box holding the name of the predicates. Figure 2 defines the problem as a whole in VEGAN's general plane.

Figure 1 below basically illustrates the rules which conclude on the relationship: "person should_buy house". "person should_buy house" succeeds if the person can "afford" the "house" and the location of the "house" is suitable (location_ok). The positive conclusion on this concept is achieved by rule2. rule1 and rule3 come to a negative conclusion if "person" (^base_frame) cannot afford to buy the "house" (^value_frame) which s/he is considering to buy or if location of the "house" is unsuitable, respectively.

```
CONCLUDE person should_buy;

defrule rule2;
[ [Same ([[^base_frame]location_ok],[[^value_frame]location]) ]
  [Same ([[^base_frame] afford],[^value_frame] )]
  => 1.0 each("house",what([[^base_frame] consider_buy]))]
endrule;

defrule rule3;
[ [Notsame ([[^base_frame] location_ok],[[^value_frame]location])]
  => -1.0 each("house",what([[^base_frame] consider_buy])) ]
endrule;


defrule rule1;
[ [Notsame ([[^base_frame] afford],[^value_frame]) ]
    => -1.0 each("house",what([[^base_frame] consider_buy]))]
endrule;

endCONCLUDE;
```

Figure 1 : a Fragment of an RBFS Program

Figure 2: A VEGAN network for the problem as a whole (below)

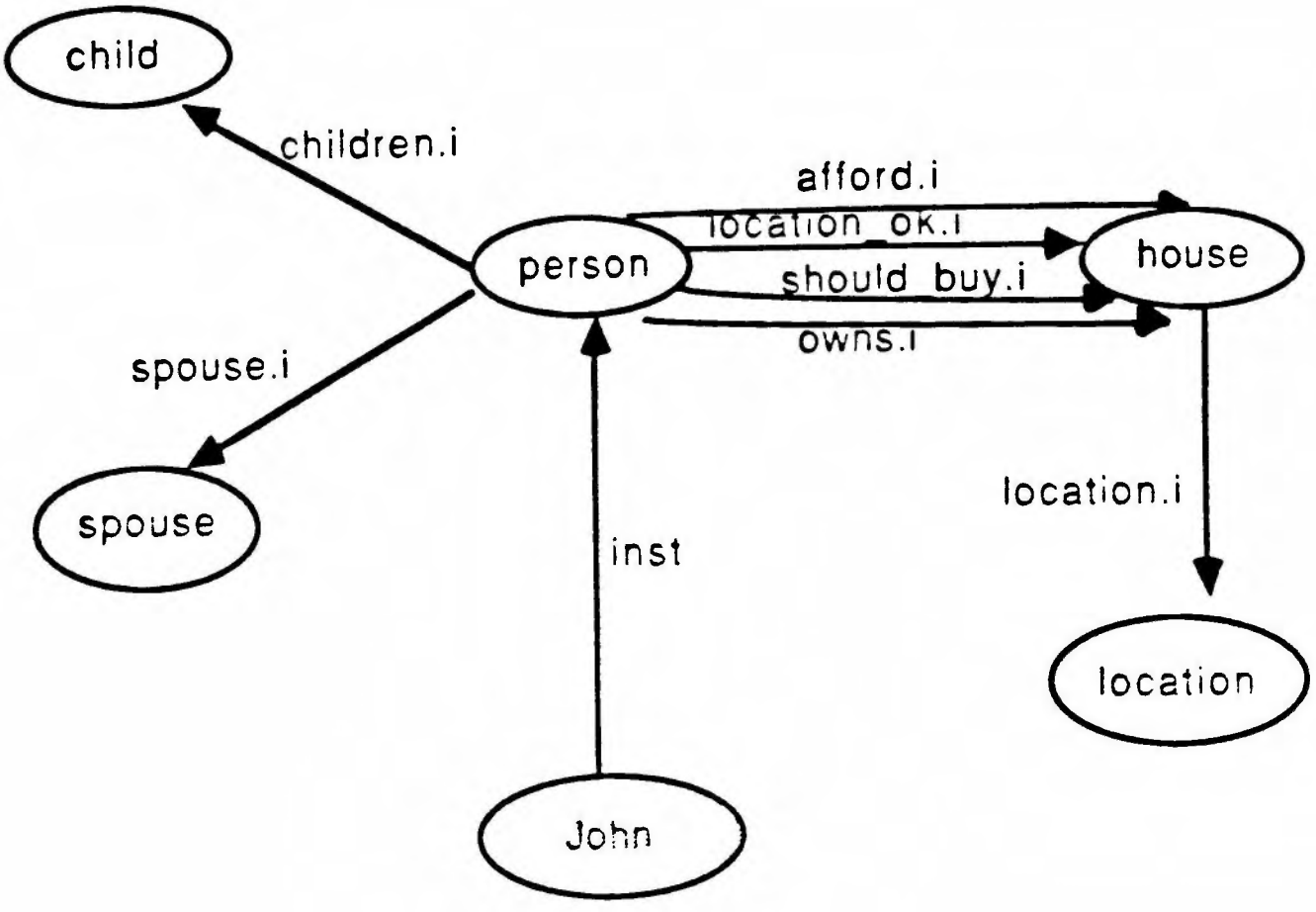


Figure 3: KET's illustration of the rules

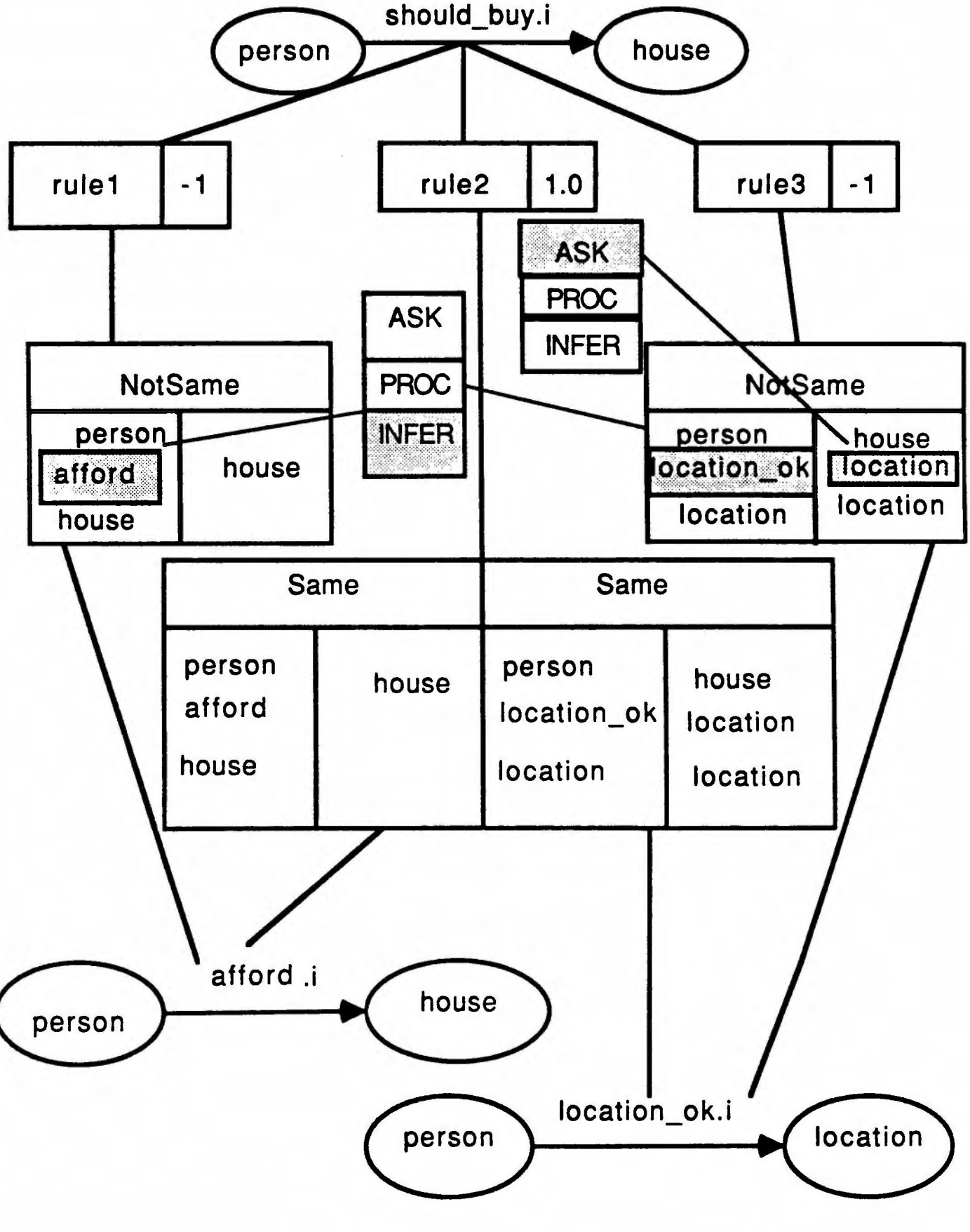
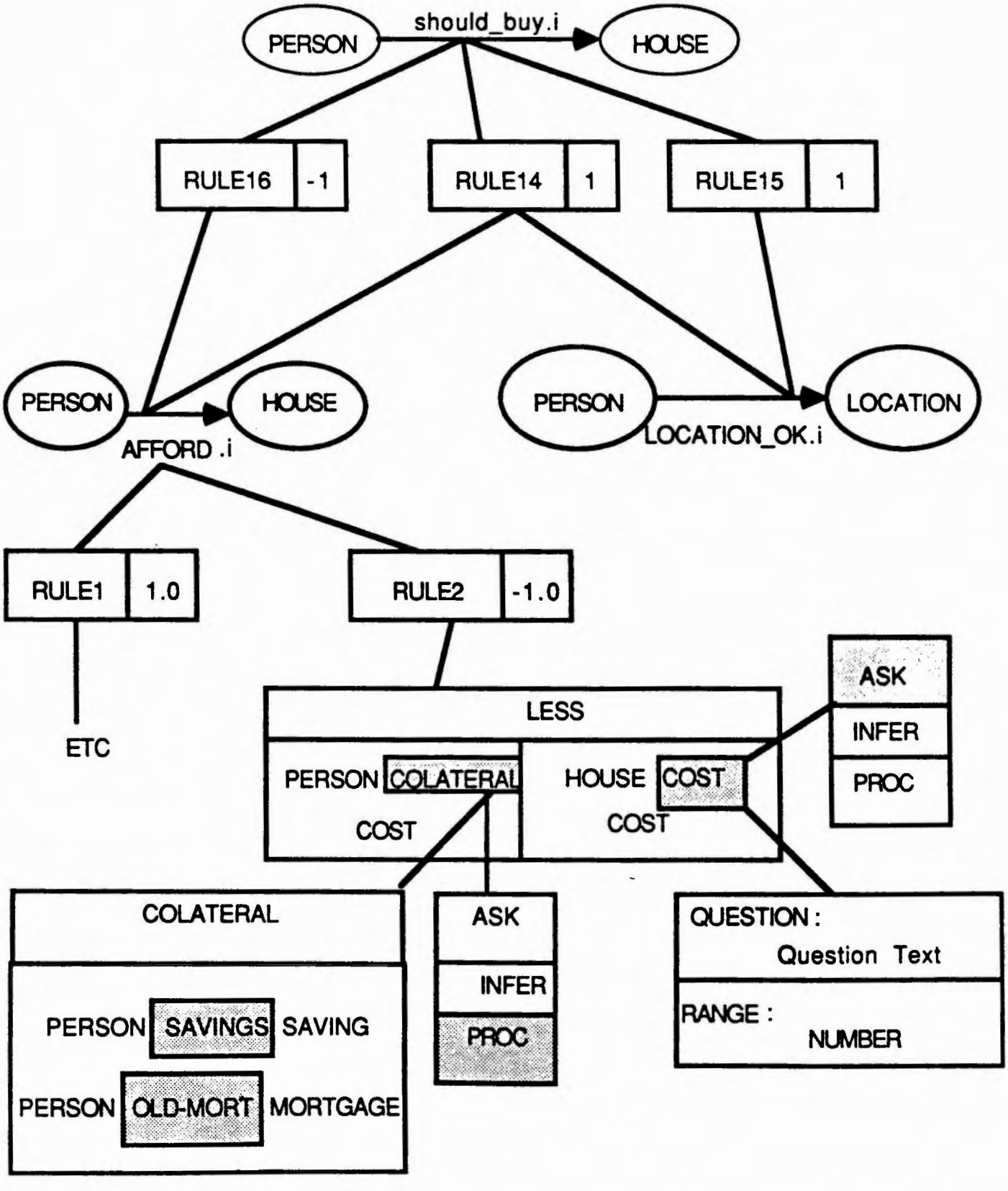


Figure 4 : KET's illustration of Ask and Procedures.



Concepts of "person" and "house" represent the generic nodes in the general plane. As demonstrated, starting from the relationship "should_buy" between "person" and "house", an expansion of the rules, CONCLUDing on "should_buy" relationship, can be invoked. Please bear in mind that the first slot, or the root of the tree does not have a parent rulename. This slot(root) is the highest level question that the client could ask. eg. in the house buying example one could ask whether a person can afford to buy a house. But the highest level of inquiry is if the person should_buy a house. ie. "person afford" is used by the " person should_buy" relationship. Therefore at the root level, the specific slot is the one which is not referenced by any other rules and the value of which is not extracted by direct data input(by ask) or procedures.

So graphically the slot("should_buy" link) is expanded to boxes embracing the rule names that conclude it. These boxes in turn, could be expanded to include the representation of predicates.

The predicates contain paths whose truths are to be established. By path we mean the node-link paths in VEGAN needed to be traversed in order to reach the ultimate value-frame.

The paths in KET start with a frame and alternate with slots from then on. A path should also end with a frame name. Any slot without a value-frame needs to be evaluated and the process continues.

Functions such as 'a', 'some' and 'each' are chosen by the menu when the mouse cursor is placed on the value box. These functions determine the quantity boundries of the value frames. This is how the specific frames inherit general frames properties in RBFS, too. [5]

However the value of the slot are not always inferred by rules. So the expert should first determine how to obtain the value of slots in newly created rules ie. determine the facet. As the frame and the slots alternate in specifying a path, the system is able to locate the slots and allow a menu option to each of the slots. In such a menu, the expert could choose the desired facets ie. whether the value frames should be asked, inferred or calculated via a procedure.

ASK - Choosing the ask option allows the expert to design a question in order to request information from the user on the specified value frame. Choosing this option via the menu will bring up a window with appropriate prompts for the user to design a question and specify the accepted range of answers. This is illustrated in figure 4.

INFER - Infer will create a graphical reminder that the box should further be expanded to include a set of new rules and premises. The process of creating rule name and predicates, described above, can then be followed by the user. This is illustrated in figure 3.

PROC - Proc. Indicates that the expert should write a procedure in order to evaluate a slot. The user should also specify the paths used in his/her procedures so that their derivation methods could be specified, too. This is illustrated in figure 4.

When the rule inputs are completed the contents of the windows are saved. The expanded rule window is then closed, leaving only the rulename box. The user can expand the rulename box to see the contents of the rules by an appropriate option in the KET menu. The two figures 3 and 4 show an animation of the graphics of KET so described.

### 3.1. KET MENU:

The menu is implemented in the C language. The menu program interacts with the POP11 procedures that act as editing functions of KET. These functions offer the user a set of actions which could be performed on boxes containing the appropriate concepts. Similar options are provided for the links between the boxes (not to be confused with the links in VEGAN. Links in KET illustrate the relationships between the boxes and do not have a name). They include commands such as: addition, deletion, save, load or renaming. Updating the boxes or their connections will automatically update everything associated with them.

The content of the boxes determine their shapes without requiring the user to remember numerous menu options. The data on the boxes and the links are stored in arrays which are updated every time the user saves the graph. Apart from the editing functions, KET's menu offers the user extra options to view the graph. These viewing operations include zooming in and out of the graph and also locating

and focusing on certain boxes and their neighbouring associates.

When an option is chosen on the menu, the system displays an appropriate message in order to assist the user in completing their task. The graphical package used in KET is suncore.

**4. Summary :**

KET is basically a rule eliciter with RBFS as its underlying knowledge-base. RBFS is basically an Expert System Shell which uses the Frame-Based representation in the form of semantic networks to incorporate one kind of deep reasoning. VEGAN visually allows generation of semantic networks.

RBFS also uses the rule based representation which was formerly written in a POP11 fashion. This would cater for the shallow reasoning of the application field. KET allows the user to write an Expert System bypassing the technicality of RBFS. It presents the users with notions of rules, certainty factors and predicates with proper guidance in a convenient fashion. The representations are implemented after suggestions from the real experts in various fields. The theoretical background of KET is supported by theories in knowledge representation and knowledge elicitation fields.

**5. Future_Developments :**

The system will be upgraded in order to check the rules for loops and also for rule redundancies. It is possible that the system could benefit from rule induction, too. This feature could be implemented with the aid of VEGAN's specific plane and rules so applied and written in KET. These rules and their related token nodes can then be generalised and grouped together to form the general plane in VEGAN. This will relieve the users from worrying over domain conceptualisation. We will, therefore, complete KET by fully exploring the field of rule learning.

The system should be able to assign appropriate rule identities to the premises. This can be automatically generated by increasing the numbering of the rules as premises are added. The system should be able to adjust its numbering as a consequence of deletion or addition of rules as the Expert System expands.

The system has been discussed among the colleagues who have had experience in building Expert Systems. This will help in achieving our goals through reliable feed backs, ie. experts. One useful feature so envisaged is reminder of conditions of rules or indeed the rules themselves which the expert might have omitted. Further checking on facets of value-frames is also essential for the sake of consistency of data derivation.

In addition to KET, RBFS is undergoing constant development, too. One major change is the abolition of the predicates. The common factor in the rules between RBFS and the new version (GIL) will be the frame - slot path about which a conclusion is to be drawn. Change of KET in order to follow suit will not be much of a hindrance.

**6. Conclusion :**

In this paper we have described a graphical system KET that we feel would be a useful tool in knowledge elicitation. With this system the expert can work independent of the knowledge engineer for a substantial part of the process. The role of the knowledge eliciter would be limited to that of initial explanation of KET's functions to the expert. The authors believe that for the very same reason automation of knowledge elicitation would be a deceiving terminology to use. Computers are a long way away from being 'really friendly'.

Nevertheless, the contribution of KET to knowledge elicitation is significant to this extent. It allows the expert to experiment with different approaches to his/her subject area without inter-personal pressures. [2] Also the key to effective knowledge engineering is to span the relevant domain by encouraging the expert to generate concepts and their inter-relationships. On-line analysis is necessary for this and it is beyond the capabilities of most people. [2]

**7. Acknowledgement :**

We would like to thank Tim Barber and Gary Marshal (the creators of RBFS) for their guidance and expressing the needs in the area of Knowledge elicitation. Our thanks also go to Stephen Lea for his experience in Expert Systems applications and for his suggestions for Expert guidance. John Kellett (the creator of VEGAN) set the path for developing the graphical part of KET.

**References**

1. Hayes-Roth, F., Waterman, D.A., and eds, D.B. Lenat, *Building Expert Systems*, Addison-Wesley , 1983 .
2. Shaw, M.L.G. and Gaines, B.R., "Interactive Knowledge Elicitation," *Canadian Information Society Session*, May 1984.
3. Hart, A., "Knowledge Elicitation: Issues And Methods," *Computer Aided Design*, vol. 17, no. 9, November 1985.
4. Barber, T.J., Marshall, G., and Boardman, J.T., "RBFS - A Rule Based Frame System," *System Engineering Group Technical report* , no. 8, Brighton, 1986.
5. Eklund, P.W., Barber, T.J., and Teskey, F.N., "An AI Environment For Knowledge Based Systems Development," *Proceedings of the KBS 87 Conference.*, London, 23-25 June 1987.
6. Wilson, J.L., , G.K. Milkoudis, and Fang, H.Y., "GEOTOX: A Knowledge-Based System For Hazardous Site Evaluation," *Journal of Artificial Intelligence*, vol. 2, no. 1, 1987.
7. Steel, L., "Second Generation Expert Systems," *Proceedings of SGES* , December 1986.
8. Musen, M.A., , L.M. Fagan, and , C.H. Shortliffe, "Graphical Specification of Procedural Knowledge for an Expert System," *IEE Computer Society Workshop on Visual Language* , 1986 .
9. Davis, R., "Interactive Transfer Of Expertise: Acquisition Of New Inference Rules ," *Artificial Intelligence* , vol. 12 , no. 2 , pp. 121-157 , August 1979 .
10. Shaw, M.L.G. and Gaines, B.R., "Knowledge Engineering For Expert Systems," *Proceedings of COMPINT: Computer Aided Technologies*, 1985.
11. Bainbridge, L., "Verbal Reports As Evidence Of The Process Operator's Knowledge ," *International Journal of Man-Machine Studies*, vol. 11, no. 4, pp. 411-436 , July 1979.
12. Kellett, J.M., "VEGAN - A Visual Editor for the Generation of Associative Networks," *System Engineering Group Technical report*, no. 12, Brighton , 1986.
13. Barr, A. and Feigenbaum, E.A., *The Handbook of Artificial Intelligence*, 1, HeurisTech Press, Stanford, 1981.
14. Laurent, J.P., Ayel, J., and Thome, F., "KEE, Knowledge Craft, Art.," *The Knowledge Engineering Review*, vol. 1, no. 4 , November 1986 .
15. Sowa, J.F., *Conceptual Structures - Information Processing in Mind and Machine*, Addison-Wesley, Reading, Massachusetts, 1984.

# A Naturalistic Methodology for Knowledge Engineering

August 1987

R. J. (Bob) Hardiman
Technical and Management Development
Mail Point 214
IBM United Kingdom Laboratories Ltd
Home Farm, Leigh Road, Eastleigh, SO5 4EU
Tel +44 (0)703 620911 ext 219

This paper outlines a new approach to the emerging discipline of knowledge engineering. It describes the new method and relates it to a methodology for naturalistic human inquiry developed by Lincoln and Guba (85) and shows the relevance for people involved in knowledge engineering.

# Outline of Knowledge Engineering

## Introduction

Knowledge Engineering (KE) is the name given to the process of eliciting human expert knowledge, from a human expert, and encoding it into a form acceptable to a new type of computer program; the "Expert System".

## Knowledge Engineering

There are two aspects to the creation of expert systems:

- The technology of inference mechanisms and the syntax of rule languages
- Knowledge engineering

  There are three parts to knowledge engineering:

  - The human process of knowledge elicitation and representation in some formal, propositional form
  - The technical process of coding propositional knowledge for entry into the computer
  - Checking the resulting expert system for "validity".

The technical process is a well researched and moderately mature discipline although much work is still being done.

Similarly well researched is the process of encoding knowledge; a wide variety of formalisms have been suggested and used (Sowa, 83).

However, the human process of eliciting the knowledge and making it plain both to the expert and to the users of that knowledge is largely un-addressed. In an extensive literature search no references other than an excellent (and much-quoted) paper by Margaret Welbank, and a paper and more recent book by Anna Hart have been found.

The rest of this paper will address this human problem, leaving the technical issues on one side.

# Naturalistic Inquiry and its Relation to Knowledge Engineering

## Forms of Human Inquiry

Human inquiry takes many forms, from academic philosophical, psychological and sociological research to the more prosaic investigations of individuals exploring they way they and their peers operate in the world.

Until recently most academic human inquiry used scientific methodology drawn from a positivist philosophy. Such research produces laws and theories, relies heavily on measurement, and on the subsequent application of mathematics and statistics to those measurements in order to make "sense" of the data acquired by the research.

More recently it has been suggested that human inquiry based on such methodologies is alienating and produces only *description* instead of *understanding*.

Many alternative approaches to positivist, academic human inquiry have been suggested, by many writers, some radically hostile to positivism, some less so. It appears that a new dialectic is emerging that will subsume both positivism and it's more radical alternatives, and which has been called "post-positivism" (Lincoln and Guba (85)). This new paradigm of inquiry has been extensively documented by Heron (71), Reason and Rowan (81), Lincoln and Guba (85), Cohen and Manion (80), Polkinghorne (81), and many others. Lincoln and Guba describe a "post-positivist" inquiry methodology which they call "Naturalistic Inquiry", which I have modified to provide a framework for knowledge engineering.

The major characteristic of any human inquiry is that people seek to increase their knowledge of people. The source of the knowledge may be far removed from the sink of the knowledge, as when a market researcher sends out a 1000 questionnaires to be completed; or completely merged, as when someone enters therapy to learn about themselves and what makes them act the way they do. Similarly, the reason for the interest may be academic, pragmatic, commercial or intensely personal. The unifying thread in all this inquiry is that seeker hopes, in some sense or the other, to discover some sort of truth or understanding -- a truth or understanding that was not immediately obvious.

## Knowledge Engineering As Human Inquiry

Knowledge engineering is a form of human inquiry, and a form especially suited to the naturalistic paradigm. In it we are seeking to discover the exact form of the expertise, and to represent it in a formal, computable, but at the same time *comprehensible* form. It is likely that the expertise is, before the inquiry, unarticulated (and perhaps thought to be inarticulable), and possibly the form of the expertise is hidden even from the expert.

Knowledge engineering involves three experts. The first is the domain expert, the person who is the expert in the field for which we are building the expert system. Another is the expert in knowledge engineering; the person who is trained in using the naturalistic knowledge engineering methodology and in using the technology of expert systems. But the most important is the potential system's user -- the person who is experienced in the use of the expertise, the person who actually has the problem. Failure to consider this party to the system is the cause of system not being used; of the many expert systems that are merely solutions in search of a problem.

Our needs from the inquiry are:

- To explore the expertise -- in Lincoln and Guba's terms "the focus of the inquiry is the problem" (to make the expertise more freely available, for instance)
- To ensure that the findings and interpretations of our inquiry are trustworthy
- To ensure that the inquiry is well-bounded (doesn't go on for ever)
- To ensure that the findings of the inquiry are relevant, timely and useful to all the respondents of the inquiry

Use of a modified form of the Lincoln and Guba methodology for naturalistic inquiry can not only guide the knowledge engineering process to ensure that these criteria are met but also give a methodological and philosophical framework for justifying the methods used and for establishing the trustworthiness of the process undertaken.

## Epistemology of Knowledge Engineering

Expert knowledge is acquired slowly over a long period of time. Frequently its development goes unnoticed: many experts find it difficult even to acknowledge or recognise their own expertise. Collins (85) describes two models of learning which he calls the "algorithmic model" and the "enculturational model". These learning models relate to the acquisition of propositional and tacit knowledge. Heron (1981B) outlines three forms of knowledge, which he calls propositional knowledge -- "knowledge of facts or truths ... entirely language-dependant", practical knowledge -- "knowing how to do something ... a skill or proficiency", and experiential knowledge -- "knowing an entity ... in face-to-face encounter and interaction.". It seems to me that there is a close parallel between Heron's practical knowledge and Polanyi's (1967) tacit knowledge -- maybe the concept of tacit knowledge embraces both Heron's practical and experiential knowledge. Certainly the factual or propositional knowledge is acquired by Collins' algorithmic method, and the other forms during his enculturational process.

I suggest that the differentiating component of expert knowledge is that it involves a large proportion of all three kinds of knowledge. The job of the knowledge engineer is to aid the process of mutual understanding of each of these kinds of knowledge in all of the parties involved in the inquiry -- the putative user of the expert system, the expert who's knowledge will be embodied in the system, and the management who bear responsibility for the successful installation and running of the system.

Only if *all* the dimensions of *everyone's* knowledge are explored, can the system hope to be successful, since the purpose of the system is to bridge the gaps between the knowledge that the expert has and the knowledge that the user has.

At our current level of technical ability I suggest that an expert system will be unable to contain experiential knowledge -- Collins' enculturational knowledge -- and that that knowledge should already be shared by both users and experts (the "common sense" of the cultural group).

Expert knowledge is usually tacit knowledge, acquired experientially over a long period of time. The expert may have been trained by formal (that is, positivistic) training methods a long time ago, but at the end of that training certainly wasn't an expert -- probably was not even considered (by self, peers or superiors) as "domain competent" -- *expertise* (that is, the addition of tacit and experiential knowledge) would be acquired by:

- Enculturation within the society of competent practitioners

- Extended experience with the *use* of the knowledge

Many experts are unable to express their knowledge in clear, propositional form -- it is in the nature of *tacit* knowledge that it can't be expressed propositionally -- and they have often forgotten the propositional form of the knowledge that they had as students. However, they can usually be helped to express the knowledge in a heuristic form and to develop new propositional knowledge that can be encoded for use in a computer system.

It is important to realise, however, that the knowledge engineering process shouldn't stop there. Whilst it is obviously vital to find propositional knowledge which can be encoded and entered into the computer to form the expert system, it is also important to explore the practical and experiential knowledge of both the expert and the system's users as the system will implicitly assume that these are similar. It is for this reason that most successful expert systems do not attempt to help people who are entirely naive in the domain of the system, but are for those who are domain competent to extend their abilities.

## Modified Methodology For Knowledge Engineering

Lincoln and Guba suggest 14 corollaries for research method leading from axiomatic differences between positivism and post-positivism:

1. Natural Setting
2. Human Instrument
3. Tacit Knowledge
4. Qualitative Methods
5. Purposive Sampling
6. Inductive Analysis
7. Development of Grounded Theory
8. Emergent Design
9. Negotiated Outcomes
10. Case Report
11. Idiographic Interpretation
12. Tentative Application
13. Focus-Determined Boundaries
14. Trustworthiness

In this section I will visit each of Lincoln and Guba's 14 points and examine them in the light of the requirements of naturalistic knowledge engineering.

### Natural Setting

Knowledge engineering should largely be carried out in the setting in which the activity normally takes place. To do otherwise robs it of the context in which it happens. It is vital that the KEs spend time with both the experts and the users, gaining an understanding of what they do and the environment in which they do it.

It may be necessary to interview away from the workplace, but it is important that all the inquirers understand that discussing an activity is no substitute for *experiencing* that activity in its normal context. Further, it is important to realise that the context is changed by the inquiry process, and that it may be necessary to spend some time in the environment before the disturbing effect of the inquiry dies down. Lincoln and Guba cite Jackson (1968)

> ... in his year-long study of a California classroom -- one in which he sat virtually every day -- even his sneezes continued to draw attention until the end of the year, although no-one attended to the sneezes of any of the "regular" members of the class.

## Human Instrument

Positivist human inquiry uses various forms of instrumentation designed to give objective descriptions of the inquiry subjects. These instruments may be mechanical, intellectual or mathematical.

Lincoln and Guba argue that the "human instrument" (people) is both necessary and sufficient for human inquiry. I believe that people need more introspective ability than is "normally" found and that this corollary should read "*trained* human instrument". This view is supported by Torbert in his article "Empirical, Behavioural, Theoretical and Attentional Skills Necessary for Collaborative Inquiry" in (Reason and Rowan 81).

Whilst teaching knowledge engineers all over the world, during several knowledge engineering projects, and during other forms of human inquiry (co-operative inquiry -- see Reason and Rowan) I have found that people need training to be effective inquirers, if they are to be able to give facilitative attention to both experts and users, and if they are to keep the inquiry as free as possible of "the biases, motivations, interests, and perspectives of the inquirer" (Lincoln and Guba 1985).

Other instruments are also available. Kelly's (1955) repertory grid has proved useful as a memory jogger and to expand the explored dimensions of the expert's and user's thinking (Shaw, Boose, Hart, Welbank), as have Eden, Jones and Sims' (1983) cognitive maps. Probably the most useful tool is a prototype system -- both experts and users have little problem in criticising the system and describing how it doesn't match their reality, even where they might have had trouble describing that reality in the first instance.

Boose (85) has developed a repertory grid system based on the work of Shaw (81) and Keen and Bell (81) in which the system guides the expert in the elicitation of a repertory grid. In this system, the grid, per se, is not important -- what is important are the implied priorities expressed by the expert. These are reflected back during the consultation for refinement, correction and clarification. At the end of the session the system will convert the knowledge expressed in the grid into heuristic, propositional form; directly into the rule language for the expert system shell being used. Whilst Boose recognises several limitations in the system, the expert system so generated serves as a prototype for criticism, as described above.

I have developed a similar system which runs on an IBM PC and which aids in the elicitation of repertory grids both by the triadic method (Kelly) and the Dyadic method (Keen and Bell) as well as allowing users to just enter extra elements and constructs as they occur to them.

Another method of developing prototypes for criticism is to use implementations of Quinlan's (1979) ID3 algorithm. The program analyses sets of examples provided by the expert (past cases who's outcomes are known) to produce an optimised decision tree that would produce the same results as the past cases. The program is useful in finding conflicting data and showing when too few attributes of the situation are being considered to be able to fully discriminate between differing cases. The result of this process has been called an expert system, but it's chief benefit seems to be the organising effect it can have on data, and it's ability to prompt experts and users to consider attributes of the problem that they might have overlooked.

I have used another PC-based tool which can take tables of examples and build decision trees, can build the trees direct, allows users to graphically navigate those trees, and finally to produce production rules for uploading to the mainframe expert system shell (IBM's Expert System Environment). Whilst our experience

has not led us to believe that this tool is in any way a panacea for all KE projects, it certainly has led to useful prototype systems from which development can then proceed.

## Tacit Knowledge

The term was coined by Polanyi (1958, 1967) to describe those abilities that we have but which we can't articulate. In many areas we acquire knowledge in propositional form, but the knowledge becomes tacit as we gain experience and forget our lessons. In these areas learners and teachers are able to articulate the knowledge in propositional forms but competent practitioners who are *not* teachers have only tacit knowledge. A learner driver can probably remember all that his instructor has told him, but is not rated as competent. Having passed the driving test and driven for several years, the driver is competent and the skill of driving has become automatic. But when the competent driver is called upon to teach a teenage offspring to drive, he finds that his driving skill has become tacit, and extremely difficult to articulate -- much thought has to be given to either remember what *his* driving instructor taught him, or to attempt to articulate what he now does.

| | Propositional | Tacit |
|---|---|---|
| Neophyte | yes | no |
| Practitioner | ?no | yes |
| Teacher | yes | ? |

Figure 1. Dimensions of Knowledge

Collins (1985) shows that tacit knowledge plays a very wide part in expertise even in areas, such as conducting natural science research., where it might be thought that all the knowledge was propositional and recorded.

The KE Project should build on and utilise the "tacit" knowledge of those involved. Knowledge engineering should be a collaborative inquiry (Heron, 1981a) and that the whole thrust of that inquiry is to make the expert's tacit knowledge plain, and to convert it to a propositional form.

Knowledge Engineering is quintessentially the process of

- Exploring tacit knowledge and converting it to propositional knowledge, encoding the propositional knowledge and entering that into the computer
- Exploring the tacit knowledge (and the *culture*) of both the experts and the users in order to ensure that the system is consonant with that culture and makes no unwarranted assumptions.

## Qualitative Methods

These are such methods as interview, repertory grid (Kelly, 55), cognitive mapping (Eden, Jones and Sims, 83), and observation, as opposed to statistical analyses of questionnaire returns and the quantitative methods of (for instance) Osgood, Suci and Tannebaum (1957).

Lincoln and Guba argue that

> Qualitative methods are stressed within the naturalistic paradigm not because the paradigm is anti-quantitative but because qualitative methods come more easily to the human-as-instrument.

They go on to emphasise the absence of an anti-quantitative stance in their paradigm. This is reasonable but insufficient. To argue for quantitative methods simply because the human researchers find them easier to manage is not enough. However, there are other arguments to support the use of qualitative methods, the most important of which are that people find greater meaning and understanding in the results obtained from qualitative data and that statistical analyses serve only to enhance the alienation outlined above.

With expert systems the use of qualitative methods is intrinsic to the nature of the system that we are developing. We are seeking to capture a sense of the expert's tacit knowledge and experience. We also need the heuristic knowledge; the tid-bits of information garnered over long experience. These can only be gathered in face-to-face encounter, with interview and careful observation.

Quantitative methods may be preferable for expressing knowledge which is susceptible to such treatment. The end results of computer system derived from such knowledge are the algorithmic systems that currently constitute most of our uses of computers. However, expert systems are being developed precisely to try to handle those classes of problems that can't be described by a neat (or not so neat) algorithm. These problems are expressed heuristically, and the expert and knowledge engineer require less of a *description* of the knowledge domain, than an *understanding* of it.

The qualitative methods used in our framework for knowledge engineering and which are taught in the class "The Human Side of Knowledge Engineering" (Bell, 1987a) are:

- Open Interview

  This interview is used in the early stages when the inquiry team are starting to explore the initial territory. Bell suggests that these open interviews are used to

  - to allow time for establishing trust ...
  - to establish understanding -- so that people have similar expectations of the project and mean the same thing by the same words
  - to establish the wide context of the project ...

- Focused Interviews
  - Case Analysis
  - Critical Incident Analysis
  - Protocol Analysis
  - Repertory Grid Analysis
  - Cognitive Mapping
  - Twenty Questions

- Observation

## Purposive Sampling

As Lincoln and Guba say, *all* sampling is done with *some* purpose in mind. However, sampling done in research in the conventional paradigm is usually looking for *representative* data; data that will go on to support general theories. These data are unlikely to say what any particular person or group of people will do at any one time. To discover that, one has to observe those people to see what they are actually doing.

So, in the social sciences, old paradigm research ignores specifics and looks at generalities, attempts to minimise the effect of the observer, and produces probabilities of behaviour. Naturalistic inquiry admits to the effect of the observer, attempts to reconcile it with the data, and primarily looks at the actual situation existing *here and now*, as opposed to the general rules.

Thus Lincoln and Guba speak of sampling for some other purpose than to calculate generalities or to look for representative data. Possibilities that they include are:

- sampling extreme or deviant cases
- sampling typical cases
- maximum variation sampling
- sampling critical cases
- sampling important (in some sense) or sensitive cases
- convenience sampling

To these I would add

- sampling interesting cases (interesting sampling)
- sampling the data that we have (pragmatic sampling)

In addition, we need to consider that in old paradigm research the research design is planned before the research begins, hence the sampling techniques are also designed ahead of time. In knowledge engineering it is frequently the case that at the outset of the research we are in a state of "meta-unknowing" (Hardiman, 87b) and hence, not knowing what we don't know, cannot predict the direction that the project will take. Thus the inquiry design emerges as the research progresses.

## Inductive Data Analysis

The induction-deduction debate is crucial between the old paradigm and the emergent new paradigm. Deduction is using pre-existent rules (or axioms) to predict what you will actually find. Old paradigm research uses these deductive inferences, combined with a careful search for the actual data, to verify the rules or axia. So, for instance, if we have a rule that says "red sky at night, shepherds' delight", we can deduce that if it is nightfall, and the sky is red, then there should be some delighted shepherds around. We can design experiments that determine the numbers of shepherds who are delighted by red-skys-at-night, and verify the accuracy or otherwise of our rule.

Induction, on the other hand, proceeds the other way round. Having large amounts of data about the weather and the affective condition of local sheep farmers and their workers we can sift this data to see if any pattern emerges. If we do indeed perceive a pattern we do so through the process of *induction*. The process will give us a general rule that we may choose to use. It will tell us nothing of *why* the shepherds are delighted (do they appreciate the aesthetic beauty of the sunset, does it signify the continuing presence of the

God of sheep farming, does it portend good prices at the next sheep market?). To discover that we need to talk to the shepherds, preferably without pre-conceptions.

In knowledge engineering we start (as much as possible) without preconceptions -- or at least we take steps to identify and acknowledge the preconceptions that, as researchers, we do hold. We collect the data (qualitatively) and then sift it looking for the patterns. This is Lincoln and Guba's inductive data analysis.

Having collected the data by qualitative methods the domain expert, the user and the knowledge engineer look at the data, analyse it inductively and produce propositional statements from it for entry into the expert system. At the same time they must produce an understanding of the context of the system, of each others' tacit knowledge and of the effect that the proposed system will have on the working patterns of the users and the experts.

All of which produces the question, "how?" -- a question not easily answered. As Lincoln and Guba say, "the art of naturalistic data processing is far from well-developed". Jill Bell and I, in teaching the "Human Side of Knowledge Engineering" course have developed a simplified version of Glaser and Strauss' (1967) constant comparative method, as described by Lincoln and Guba. Glaser and Strauss describe a four stage process for the processing of data and the development of grounded theory. The stages are

- Comparing incidents applicable to each category
- Integrating categories and their properties
- Delimiting the theory
- Writing the theory

This scheme seems to suggest that the categorization has happened before the data processing has begun, and the data are assigned to those categories. Instead we use a simpler process of unitization and categorisation derived from Lincoln and Guba. We have simplified the terminology, and refer to "chunking" and "sorting".

## Chunking

Judi Marshall (1981) is quoted in Lincoln and Guba and they suggest that she finds the task of dividing the data up into units as "absurdly easy" and call her attitude "cavalier". What Marshall actually says is:

> ... the units are fairly obvious -- you get chunks of meaning that come out of the data itself ... chunks of meaning, and you don't have to look at the individual sentences, or debate what the level of analysis is. Also, the books say, "Arrive at the categories that you will use." Well, I don't do that either, but let the categories build up ... as I put things together that *go* together.

Lincoln and Guba feel that units should have two characteristics: they should be *heuristic*, and they should be the smallest piece of information that should stand by itself. I don't see that this contradicts what Marshall is saying, and experience of analysing data in knowledge engineering projects suggests that a process akin to that that Marshall suggests is perfectly adequate.

The process that we use is to take the data (usually taped -- audio or video) and have it transcribed or transcribe it ourselves. We ask the typist to include everything; coughs, laughs, silences: these are important indications of the tacit knowledge -- of the culture of the situation. When we have the transcription, we sit and listen to the tapes again, checking the transcription, and reliving the experience of the interview. From there the transcript is broken up into units, firstly with a highlighting pen, later copied onto pieces of card. Some students in our class have cut up transcripts with scissors. Latterly we have used those ubiquitous yellow "post-its".

The chunks, as Marshall says, are usually fairly obvious and to suggest that we should prescribe unitization by words, sentences or paragraphs would be to impose an alien structure on someone else's data.

It is important at this stage to keep a note of where each piece of data comes from -- in later stages it is inevitable that we will have forgotten the context of a piece of data, and will need to refer back, either to the original transcript, or even to the tape of the interview. This process is easier if the interview is video-taped, especially if the camera is capable of recording a stop-watch with the picture.

As the transcribing is done with a computer-based word processor, we are considering using computer aids to categorising the data, but have made little progress in this direction, yet.

### Sorting

Glaser and Strauss' four stages of the constant comparative method seem to suggest that the data analyst should start with an idea of the categories for the data ("comparing incidents applicable to each category ... integrating categories and their properties"). Marshall, however, prefers to let the categories emerge from the data "let the categories build up ... as I put things together that *go* together".

I also prefer to let the categories build up, and use the method outlined by Lincoln and Guba

## Development of Grounded Theory

Grounded theory is a term developed by Glaser and Strauss (1967) to describe a theory that will:

> fit the situation being researched, and work when put to use. By "fit" we mean that the categories must be readily (but not forcibly) applicable to and indicated by the data under study; by "work" we mean that they must be relevant to and able to explain the behaviour under study.
>
> (Glaser and Strauss, 1967, quoted in Lincoln and Guba 1985).

Lincoln and Guba also quote Reason (1981) in an appreciation of Paul Diesing's book *Patterns of Discovery in the Social Sciences*:

> The information that is discovered in the field situation is used by the holist to build a model which serves both to describe and to explain the system. The model is built by (quoting Diesing) "connection themes in a network or pattern"; the connections may be of various kinds but they are "discovered empirically rather than inferred logically"; the result of this is an empirical account of the whole system. This account *explains* the system because it describes the relations the various parts have for each other, so that the relations between that part and other parts serve to explain or interpret the meaning of that part". This type of explanation is called a *pattern model* of explanation ...

In the knowledge engineering process we are looking for rules that are not necessarily true for all people at all times, but which fit our situation and which will work in our system.

## Emergent Design

As noted previously (in the section on purposive sampling) naturalistic inquiries frequently inquire into areas about which the researcher knows very little before the inquiry begins. In this case the inquirer is is in a state of meta-unknowing -- not knowing even the extent of her ignorance. In this case it is very difficult to design the whole inquiry before it begins. Instead the design of the inquiry unfolds as the inquiry unfolds. With developing knowledge and understanding the course and design of the research becomes apparent.

As the next stage of the research becomes clear it is apparent that we need to redesign our sampling strategy, collect new (qualitative) data, analyse the data inductively and produce new grounded theory. Which in turn will lead to the emergence of the *next* stage of the inquiry.

Thus the inquiry iterates around these four stages of purposive sampling, inductive analysis, development of grounded theory and the emergence of the next stage of the research design.

This iteration continues according to Lincoln and Guba until "redundancy". They suggest that redundancy could come because we have reached the boundaries of our inquiry, because we have found out what we wanted to find out, because the resources of the inquiry are exhausted, or for any other reason. Both the iterative process and its completion need to be carefully monitored: on the one hand it is possible to continue the research indefinitely, "since it will continually dredge up new questions and insights worth pursuing" (Lincoln and Guba, page 211). On the other hand, it is possible to close the inquiry prematurely, possibly because of inadequate resources, or because of some pathology of the inquirers.

## Negotiated Outcomes

In conventional research the research subjects seldom have any say whatsoever in the design, conduct or making sense of the inquiry. In some cases they may even take part unwittingly and thus not even have any say in their participation. This will not be the case with naturalistic inquiry since, in order to understand the sense, context and meaning of that being studied we have to engage with the subjects much more closely. This engagement reaches an ultimate in co-operative inquiry (Heron, 1981B), where the distinction between the research and subject fades away.

With this close engagement between researcher and subject it is no longer possible for one person or a group of people to analyse the data, make their own sense of it and publish it, or act upon it without negotiation from the other people involved in the inquiry. Hence the outcomes of a naturalistic inquiry are developed by a process of negotiation between all those involved. Space should be made for all respondents to have their say, and for all views to be expressed.

Lincoln and Guba suggest that the outcomes of any human inquiry; any *naturalistic* human inquiry, should be negotiated with all who took part in the inquiry. Heron (1981) argues:

> When I am interpreting basic axioms such as walk, talk, look, point in terms of their more complex intentions and purposes, then I need to check against the agent's version of what he was about, for a person may walk, talk, look or point to fulfil many different higher order purposes.
>
> (quoted in Lincoln and Guba)

Heron goes on to say that conventional research usually exploits people, "for knowledge is power that can be used against the very people from whom the knowledge was generated.". These are practical, ethical and methodological reasons for negotiating the outcomes of the inquiry with the respondents. The knowledge engineer has an even more pressing reason, should he need one: the outcome of the knowledge engineering process is a computer system that will only be successful if it is of use to, and is used by, the intended users. If these people are not fully a part of the inquiry, consulted with and negotiated with at each step along the way, then their recourse will inevitably be to spurn the system when completed.

I have come across many technically successful expert systems that are not used. In almost every case it was because nobody asked the users. The outcome of the inquiry was not negotiated. In one case a system was developed with the co-operation of the prime- and second-shift workers, the night-shift workers not be consulted. The completed system is used by the prime- and second-shift workers, and ignored by the night shift.

## Case Report

Following on from the negotiation of the sense making of the inquiry comes the issue of how it is to be recorded and presented for publication or for the use of others. Lincoln and Guba give three major reason for the use of the case report over the more usual scientific report:

1. The case report is ideal for providing the "thick description"
2. It is the form most responsive to the axioms of naturalistic inquiry
3. It is the ideal form for communicating with the consumer.

Lincoln and Guba argue that just as the scientific report that appears in journals suits research within the conventional paradigm, so the case report suits new-paradigm research. This is, perhaps, less the case with knowledge engineering, although it could be argued that a case report summarising the findings of the inquiry will be more accessible to all concerned than will be the internal workings of the expert system.

Nevertheless, most expert systems are all that remain to be seen of the passage of the knowledge engineering process. If those systems are couched in a representation that is truly understandable by all respondents, then maybe that will suffice, but even then the system is only a record of the uncovered knowledge, and not really of the higher level of metaknowledge, or even of the inquiry process. Records of both these areas can be of intense interest to those who took part, and may well be of interest to those who come after, as maps of at least part of the territory, even if the boundaries are still annotated with the "here there be dragons" legend.

## Idiographic Interpretation

Consequent upon the inquiry generating grounded theory -- theory applicable to this case, these people and this situation, and good to explain these data -- is the fact that the findings of this inquiry will not *necessarily* be applicable to other people, other groups, or other situations. Hence the interpretation of the inquiry results should be made in the context in which the inquiry happened. It is up to other readers of the case report to judge for themselves whether the results will be relevant to them, in their situations, by comparing the context of the inquiry with the context of their own situation.

Lincoln and Guba point out that "conventional investigators tend to reject idiographic interpretation as useless -- what good does it do to know about a single site only?". They go on to relate that stance to the traditional purpose of science -- to predict and control. If we take some other purpose for inquiry -- Lincoln and Guba suggest *understanding* -- then idiographic interpretation becomes mandatory.

So with expert systems. Systems are written for the context in which the expertise is currently used. It is in the nature of expertise that it has narrow context.

## Tentative Application

A further corollary of naturalistic inquiry, flowing from the grounded theory and idiographic interpretation is that the findings will seldom be applicable in other contexts. Lincoln and Guba suggest that the findings may not even be applicable in the *same* context at a later time, although it is arguable that if the time is different, so is the context.

This lack of generalisability leads us to take care about how we apply the findings of the research in other contexts, even to the extent of realising that what might seem to us as the *same* context as the inquiry

context is no longer the same -- we have now conducted an inquiry where no inquiry had be conducted before, we all know more than we did before and, it can be supposed, we are all likely to act differently from the way we acted before the inquiry.

It is thus that we should be tentative in the application of the results of the inquiry -- especially if that application is intended for a completely separate context; one in which those applying the findings didn't participate in the original inquiry.

Expert systems will be tentatively applied; probably more so than with many other sets of findings. It is often difficult to doubt the authoritative scientist; the machine is easier to doubt.

## Focus-Determined Boundaries

The openness of naturalistic inquiry and the emergent nature of inquiry design leads to the charge that the subject of inquiry is entirely at the mercy of the inquirers' subjectivity. Lincoln and Guba bound the naturalistic inquiry by looking at the original focus of attention.

The open, emergent nature of the knowledge engineering process requires that it be bounded in some manner, if it isn't going to go on for ever. Lincoln and Guba suggest that the boundaries of the naturalistic inquiry process are defined by the focus of the inquiry. In a recent knowledge engineering process we were looking at the feasibility of implementing an expert system to assist with a particular process in an international airline. After one pass round the cycle it became clear that the application was unsuitable: the experts used very simple knowledge for most of their task, and the complex part was beyond the scope of any commercial computer system. The focus of the inquiry was to assess applicability and within 24 hours that had been accomplished. So the inquiry was complete; bounded.

Other knowledge engineering projects may not be so clear. It is important that early on in the project the boundaries are explored and criteria for completion established. However, given the emergent nature of the design it is necessary to keep those boundaries under continual review.

## Trustworthiness

The question of *validity* has long been a thorn in the side of new paradigm research. The subject is dealt with at length in Reason and Rowan. Lincoln and Guba have cut through a lot of the problems by suggesting that validity is a conventional measure of the *trustworthiness* of a piece of research. They seek to look at the questions that underlie the establishment of the conventional criteria of trustworthiness -- internal and external validity, reliability and objectivity. They suggest that these questions are:

- How can one establish confidence in the "truth" of the findings of an inquiry for the respondents ... ?
- How can one determine the degree to which the findings of an inquiry may have applicability in other contexts ... ?
- How can one determine whether the findings ... would be consistently repeated if the inquiry were replicated with the same (or similar) respondents in the same (or similar) contexts?
- How can one establish the degree to which the findings of an inquiry stem from the characteristics of the respondents and the context, and *not* from the biases, motivations, interests, and perspectives of the inquirer?

In order to fulfil this requirement of *trustworthiness*, I suggest that the KE process should be subject to audit. Lincoln and Guba, writing about the process of Naturalistic Inquiry (Lincoln and Guba, 1985) suggest that

researchers involved in Naturalistic Inquiry should submit their work to outside audit, in much the same way as accountants do, or as counsellors and therapists engage in supervisions sessions.

This concept is not new in the computer industry -- we hold structured walkthroughs, design reviews and software inspections. The same process is relevant to expert systems.

The qualities required of the auditors are that they should:

- be aware of the knowledge engineering process
- not be involved in the project -- able to see the wood for the trees
- have skills as a chairman and mediator

The process of auditing involves:

- The KE preparing material for the audit session: current prototype, design description, project log (what's been done so far, who interviewed, when, etc)
- The Audit should involve everyone; users, experts, KE, management, auditor
- Audit process checks the KE process, looks for errors, omissions, personal bias, un-necessary additions

The process then provides a checkpoint before the next iteration of design or development.

## Conclusion

This methodology has been taught in knowledge engineering classes around the world (Bell, 1987b) and has been used in a variety of projects within IBM. All our evidence is that the tenets of the Lincoln and Guba methodology are relevant to knowledge engineering.

However, much work still remains to be done, both in the investigation of pragmatic tools to aid in the process of knowledge elicitation, acquisition, analysis and representation, as well as in the investigation of other techniques from the social sciences to see what else exists that will help the knowledge engineer.

I am currently in the process of looking at the phenomenological and hermeneutic aspects of knowledge engineering as part of a post-graduate research project within the School of Management at the University of Bath.

At the pragmatic end of the scale we are investigating the use of computer aids to knowledge collection and analysis. We have also recently learned of the "Seven New Methods" of qualitative data analysis of Professor Shiji Shiba, and are evaluating their relevance to this subject.

# Bibliography

Bannister, Donald; Fransella, Fay (1971) *Inquiring Man* Penguin, London

Bannister, Donald; Fransella, Fay (1977) *A Manual for Repertory Grid Technique* Academic Press, London

Bell, Jill (1987a) *Notes for the Course "The Human Side of Knowledge Engineering"* JB Associates, Winchester

Bell, Jill (1987b) *Teaching Knowledge Engineers the Human Side of Knowledge Engineering"* Proceedings of The First European Workshop on Knowledge Acquisition for Knowledge Based Systems, University of Reading, UK

Boose, J H (1984) *Personal Construct Theory and the Transfer of Human Expertise* Proceedings of the National Conference on Artificial Intelligence (AAAI-84)

Boose, J H (1985) *A Knowledge Acquisition Program for Expert Systems based on Personal Construct Psychology* International Journal of Man-Machine Studies Vol 23, Number 5

Boose, J H (1985) *Combining Expertise from Multiple Experts* Forthcoming Paper

Cohen, L; Manion, L (1980) *Research Methods in Education* Croom Helm, London

Collins, H.M. (1985) *Changing Order* Sage, London

Eden, C.; Jones, S.; Sims, D. (1983) *Messing About in Problems* Pergamon, Oxford

Glaser, B.G.; Strauss, A.L. (1967) *The Discovery of Grounded Theory* Aldine, Chicago

Hardiman, R.J. (1987a) *A Naturalistic Approach to Knowledge Engineering* School of Management, University of Bath

Hardiman, R.J. (1987b) *Education as Co-Operative Inquiry: A Methodology for Tutored Peer-Group Learning* University of Bath, School of Management

Hart, Anna (1985) *Knowledge Elicitation: Issues and Methods* Computer Aided Design, Vol 17, Number 9, January 1985

Hart, Anna (1986) *Knowledge Acquisition for Expert Systems* Kogan Page, London

Heron, J. (1981A) *Philosophical Basis for a New Paradigm* in Reason and Rowan

Heron, J. (1981B) *Experiential Research Methodology* in Reason and Rowan

Kelly, G A (1955) *The Psychology of Personal Constructs* Norton, New York

Kohak, E (1978) *Idea and Experience* University of Chicago Press, Chicago

Kuhn, Thomas, S (1970) *The Structure of Scientific Revolutions* University of Chicago Press, Chicago, Second Edition

Lincoln, Yvonna, S; Guba, Egon, G (1984) *Naturalistic Inquiry* Sage, California

Marshall, Judi (1981) *Making Sense as a Personal Process* In Reason and Rowan

Osgood, C.E.; Suci, G.J.; Tannenbaum, P.H. (1957) *The Measurement of Meaning* University of Illinois Press

Polanyi, M. (1958) *Personal Knowledge* Routledge and Kegan Paul, London

Polanyi, M. (1967) *The Tacit Dimension* Anchor, New York

Polkinghorne, Donald (1983) *Methodology for the Human Sciences* SUNY Press Albany, New York

Pope, Maureen L; Keen, Terence R (1981) *Personal Construct Psychology and Education* Academic Press, London

Quinlan, Ross (1979) *Discovering Rules From A Large Collection Of Examples* in Michie, D (ed) *Expert Systems in the MicroElectronic Age* Edinburgh University Press, Edinburgh

Reason, Peter; Rowan John (eds) (1981) *Human Inquiry -- A Sourcebook of New Paradigm Research* Wiley, Chichester

Shaw, M L G (1980) *On Becoming a Personal Scientist* Academic Press, London

Shaw, M L G (1981) *Recent Advances in Personal Construct Technology* Academic Press, London

Shaw, M L G; McKnight, Cliff (1981) *Think Again* Prentice-Hall, Englewood Cliffs, N.J.

Sowa, J F (1983) *Conceptual Structures* Addison-Wesley, Massachusetts

Waterman, D (1985) *A Guide to Expert Systems* Addison-Wesley, Wokingham

Watson, L (1986) *Earthworks* Hodder and Stoughton, London

Welbank, Margaret (1983) *A Review of Knowledge Acquisition Techniques for Expert Systems* British Telecom Research Laboratories, Martlesham Heath, Ipswich

# An Interactive Strategic Planning System.

Authors: R.V. Williams and J.R. Thomas, University of Warwick.

Date: 26/6/87

## Abstract.

IKBS techniques for Knowledge Elicitation, Knowledge Representation, Inference and Planning have been applied to many relatively low-level (tactical) domains. However, they are equally applicable to the elicitation, storage, retrieval and manipulation of high-level (strategic) knowledge.

This paper describes the design and use of an interactive computer system, intended to enhance strategic (high-level) planning for individuals and organizations.

The principal difficulties encountered in using the system have involved Knowledge Elicitation, due to the problems of mapping "informal" and "intuitive" strategic knowledge onto a formal representation scheme.

The current system is a prototype for a "Strategic Planner's Assistant".

## Keywords:

Strategic Planning, Knowledge Elicitation, Knowledge Representation.

## Introduction.

This paper describes the development of an interactive computer system intended for use in strategic planning: that is, in the selection of long-term, high-level goals and objectives for individuals and organizations.

An earlier project (Williams, 1986) provided the basis for the present work. The system developed for that project was applied to practical problem-solving, in order to assess its limitations and thus clarify the requirements for the present system.

The paper focuses upon the functional capabilities of the system, rather than upon the details of its current implementation.

This system represents an interim stage in an ongoing project, rather than a definitive version. The eventual aim is to build a Strategic Planner's Assistant.

## Motivation for the Project.

### Why Choose Planning?

Motivation for the project, and for its precursor (Williams, 1986), stemmed from an awareness of the descriptive inadequacy of previous models of human planning, and of the practical shortcomings of previous planning tools. More importantly, it stemmed from an awareness of the general inefficiency and ineffectiveness of human activity.

Observation of human activity gives support to the hypothesis that individuals and organizations are disempowered by inappropriate knowledge. They fail to achieve important goals because they lack appropriate conceptual frameworks for planning and understanding. The absence of appropriate constructs, and the presence of inappropriate constructs, causes them to misunderstand their situation. The lack of appropriate rules for planning, and the prevalence of inappropriate rules, causes them to to take ineffective or counterproductive action.

From the above hypothesis, it follows that if an improved conceptual framework for planning were to be made available, it would have a considerable impact upon the efficiency and effectiveness of human activity.

Why Choose Strategic Planning?

Planning research projects may be categorized according to their position on the strategic vs. tactical dimension. Most AI planning research to date has focused upon problems located near the tactical pole of this dimension; that is, problems which involve reasoning about methods and actions rather than about goals and values. There are exceptions to this (e.g. Carbonell, 1978; Wilensky, 1983), but they are few and far between.

However, there are grounds for asserting that research into strategic (high-level) planning should take precedence over research into tactical (low-level) planning. In any justifiable corporate operation, the selection of appropriate strategic goals, values and priorities must always precede tactical plan creation and execution. The strategic level is more critical than the tactical level; strategic errors are more costly than tactical errors. An activity which is performed without reference to a coherent strategic framework of goals, values and priorities cannot be justified on rational grounds; it takes place in an apparent motivational vacuum.

Paradoxically, the techniques required for strategic planning are less demanding than those required for tactical planning. There is less involvement with spatial or temporal relationships, or with complex resource allocation issues. This provides additional motivation for the redirection of research from tactical to strategic issues.

Objectives of the System.

The specific objective of this project was the development of a system which would assist in the following tasks:

a. For organizations: clarification of corporate goals, values and priorities; clarification of the rules and assumptions which govern the relationships between these goals; formulation of corporate policy; selection of strategic objectives.

b. For individuals: clarification of personal goals, rules and assumptions; detection and monitoring of goal conflict; selection and refinement of possible courses of action.

It is worth mentioning at this stage that the system is not intended to tackle task scheduling, robot planning or design problems; such problems require the manipulation of temporal and physical constraints, which are generally relaxed in strategic planning.

A Functional Overview of the System.

The system (see Diagram 1) elicits a problem description from the user, checks the description for consistency, examines possible plans for achieving the user's stated goals, selects an appropriate plan, and displays the plan to the user.

Having inspected the plan, the user may be dissatisfied with it and will want to change the problem description; the process of plan selection and display is then repeated, incorporating these changes.

When a plan fails, the system displays the critical elements of the problem description, so that the user can review them, and modify them if appropriate.

The elicitation/planning/display cycle is repeated until a satisfactory plan has been generated, or until the user decides to terminate the session.

A longer-term elicitation/planning cycle is involved when the user returns to the system for subsequent sessions. In this cycle, the plans accepted by the user at the end of the first session are executed in the real world; any difficulties encountered are recorded as changes to the problem description in subsequent sessions; new plans are then generated.

Knowledge Representation.

The system represents the user's problem using the following basic elements:

a. PROPOSITIONS which represent states.

b. RULES which represent implicatory relationships between states.

c. ENTITIES which represent objects (persons, organizations, etc) in the problem domain.

d. ATTRIBUTES which represent properties of objects.

All of the system's data and knowledge structures are derived from these basic elements.

The format of propositions.

The following two forms of proposition (unary and binary) are the minimum required for the description of strategic problems. The terms "unary" and "binary" refer to the number of arguments (one and two respectively) in the equivalent expressions in first-order logic.

The terms of propositions have been sequenced to conform to their natural language equivalents, for the sake of intelligibility to the user. Infix notation is used. However, their equivalents in formal logic notation are given here for clarity.

a. Unary propositions: (ENTITY PREDICATE DESCRIPTOR)

e.g "I am employed"

in logical notation, this form is equivalent to:

predicate(entity)

e.g. employed(I)

N.B. The propositions elicited by the Repertory Grid technique (e.g. Gaines and Shaw, 1980) are all unary.

b. Binary propositions: (ENTITY ATTRIBUTE PREDICATE VALUE)

e.g. "John employer = ICL"

in logical notation, this form is equivalent to:

predicate (attribute(entity),value)

e.g. =(employer(John),ICL)

In binary propositions, the value may either be expressed as a literal, or as another entity-attribute pair, for example:

"company assets > (company liabilities)"

in logical notation:

greater-than(assets (company), liabilities (company))

Some attributes, for example "age", are single-valued; others, for example "parent", are multiple-valued. The present formalism distinguishes multiple-valued attributes from single-valued attributes by the use of the predicate "include", as indicated by:

"company creditors include IBM"

include(creditors(company), IBM)

A minimal set of predicates for binary propositions are:
equal(s), include(s), =, >, <, >=, =<.

Meta-attributes of propositions.

Each proposition has the following additional (meta)attributes:

TRUTH-VALUE which may have the values:
+1 (true) or -1 (false).

DESIRABILITY which may have the values:
+1 (desirable) or -1 (undesirable).

Numeric values are used for Truth and Desirability, because a future version of the system may incorporate reasoning with uncertainty.

Top-level propositions also have the attribute:

IMPORTANCE which may have values in the range:
0 (totally unimportant) to 9 (very important).

Numeric values are used for Importance, because the planning procedures need to evaluate the benefit of achieving particular subgoals. Benefit is calculated by subtracting the sum of the importance values of top-level goals negated by a subgoal from the sum of the importance values of top-level goals supported.

The Format of Rules.

This is similar to the standard production rule format:

(if X then Y)

A rule has the following attributes:

CONCLUSION which is a proposition.

PREMISE which is a conjunction of propositions.

CERTAINTY which has a value of +1 or -1.

A proposition may appear in the premise or conclusion of any number of rules.

If a proposition does not appear in the premise of a rule, it is a "top-level goal".

If a proposition does not appear in the conclusion of a rule, it is a "primitive subgoal".

If a proposition appears in both premises and conclusions, it is an "intermediate subgoal".

From the viewpoint of two-valued logic (in which truth-values and certainties may only be +1 or -1), a top-level or intermediate proposition is TRUE if it appears in the conclusion of at least one supporting rule (in which certainty = +1) whose premise is true, and if it does not appear in the conclusion of any negating rule (in which certainty = -1), whose premise is true.

Current versions of the system use two-valued logic; future versions will incorporate reasoning with uncertainty.

The IMPORTANCE of a proposition is a function of the importance of the conclusions of the rules in which it appears as a premise.

Similarly, the DESIRABILITY of a proposition is a function of the desirability of the conclusions of the rules in which it appears as a premise.

Knowledge Elicitation.

In the current version, all of the knowledge involved in planning is elicited from the user during the session. A future version will permit the pre-loading of domain-specific knowledge.

The knowledge elicitation process is driven by a search for relationships between propositions, using a version of the "laddering" technique (Hinkle, 1965), and proceeds in the following stages:

Eliciting the user's explicit goals.

The user is first asked to declare his/her goals and values as a list of propositions, each of which represents a goal to be achieved or a value to be preserved. For example:

(I am working)
(I am independent)

Eliciting the user's superordinate goals and values.

The user is then asked to declare the "desirable consequences" of achieving the goals and values elicited in the previous step. For example, a desirable consequence of "I am working" might be "I am earning". "I am earning" is a superordinate of "I am working"; "I am working" is a subordinate of "I am earning".

The next step is to ask the user to declare the desirable consequences of the superordinate goal. This "upward-laddering" procedure continues the user declares a goal which has no superordinates (that is, it is seen as an end in itself). This goal is marked as a top-level goal.

The process is repeated for the other goals (e.g. "I am independent"), thus building a set of "goal trees" or hierarchies.

As each successive top-level goal is declared, the system attempts to elicit the relationships between the newly-elicited top-level goal and those already stored. The possible relationships are:

equivalence: the new goal implies, and is implied by, an existing goal.

opposition: the new goal negates, and is negated by, an existing goal.

superordination:
the new goal is implied by an existing goal, or
the new goal is negated by an existing goal.

subordination:
the new goal implies an existing goal, or
the new goal negates an existing goal.

independence:
the new goal has no implicatory relationships with existing top-level goals.

If the new top-level goal is independent, the user is asked to indicate its importance relative to the other top-level goals, by assigning a numeric "weight" to each of the goals.

Eliciting the subordinates of the explicit goals,

Having elicited the superordinates, the system then asks the user to declare the subordinates of the original goals (that is, the conditions or circumstances in which the achievement of these goals is possible or impossible). For example, "I am working" is possible when "I am employed", and impossible when "I am ill".

First, the system elicits the negative subordinates of the goals (i.e. the conditions under which the goal cannot be achieved).

Then, it elicits the positive subordinates by asking the user to specify the circumstances in which the goals in the problem statement can be achieved.

A particular goal may have more than one subordinate. It is necessary to determine whether the preconditions are conjunctive (that is, the goal is only achieved/negated if all of them are true) or disjunctive (the goal is achieved/negated by any one of them).

Having elicited the immediate subordinates, the subordinates' subordinates are elicited. This "downward-laddering" process may continue indefinitely. In practice, it is more useful to build a fairly modest goal network in the first elicitation session. A large network is difficult to debug.

While eliciting subordinates, it is also useful to elicit alternative superordinates (independent higher-level goals which typically conflict with the existing superordinates of the goal under consideration), for example:

SUPERORDINATE GOAL: "I am employed"
SUBGOAL: "my employment-type = employed-by-others"
ALTERNATIVE (NEGATIVE) SUPERORDINATE GOAL: "I am independent"

Upon encountering an alternative superordinate, the downward-laddering search for subordinates is suspended, and the upward-laddering search for superordinates is invoked. The search for subordinates restarts when the alternative superordinate has been traced up to a top-level goal.

The search for subordinates terminates when all of the goals in the problem statement have been traced down to primitive subgoals (that is, subgoals for which no further subordinates can be elicited).

Reviewing and Editing the Problem Description.

When the goal network has been elicited, the Knowledge Base Editor is invoked, and the user is encouraged to examine the goal network, and to modify it if required. New superordinates and subordinates can be linked to any goal in the network. Modification is subject to monitoring by validation procedures. The coherence of the network is maintained: new superordinates are traced up to top-level goals, and new subordinates are traced down to primitive subgoals.

Detecting Structural Anomalies in the Problem Description.

Upon exit from the Editor, the system checks the goal network for certain structural features which preclude effective planning. These anomalies must be repaired by the user before the planning procedures may be invoked. The anomalies are:

a. Circular Implications, in which a proposition is its own consequent, as in:

A -> A or A -> B -> A

or in which a proposition implies its own negation, as in:

A -> not-A or A -> B -> not-A

b. Contradictions, in which a proposition both supports and negates a particular goal, as in:

A -> B and A -> not-B

Planning.

When a problem description contains no anomalies, control is transferred to the planning procedures.

The planning procedures are driven by a search for a set of primitive subgoals whose satisfaction optimizes the benefit (in terms of the user's priorities) of the top-level goals achieved.

Plan effects are represented as assertions to the system's database, that is, as values bound to particular attributes of particular entities.

Appendix 1 shows the planning procedures in action.

Plan Generation:

Plans are examined and selected by backward-chaining through the network of propositions, starting with the highest-priority top-level goal.

Whenever the goal under consideration has subordinates, the following procedures are invoked. This invocation continues recursively until each of the top-level goals has been either achieved or negated.

Negation of Negative Subordinates: a goal whose negative subordinates are true cannot be satisfied. Therefore, the first procedure invoked in the attempt to achieve a goal is one which attempts to negate the goal's negative subordinates. If there is more than one conjunction of negative subordinates, each conjunction must be negated.

Achievement of Positive Subordinates: if the negative subordinates are non-existent or have all been negated in the previous step, the system will then attempt to achieve the goal's positive subordinates. If there is more than one conjunction of positive subordinates, it is sufficient to achieve only one of the conjunctions.

The Evaluation Functions.

Where there is a choice of subordinate propositions or conjunctions to be achieved or negated, the system evaluates the possible consequences (in terms of the top-level goals) of achieving or negating each subordinate.

The evaluation may involve:

a. evaluating each subordinate to determine whether it is already true or false;

b. tracing the explicit implicatory links from each subordinate up to the top-level goals;

c. tracing the implicit links from each subordinate to other propositions which refer to the same entity and attribute, and thence up to the top-level goals.

The subordinate whose achievement/negation has the greatest apparent benefit (in terms of the importance of top-level goals achieved, minus the importance of top-level goals violated) is selected.

Display of the Plans.

The plans are then shown to the user for approval. The Knowledge Base Editor enables the user to navigate the network of relevant goals, and to examine explanations for the choice of subgoals at any particular point in the hierarchy.

If the problem description is incomplete, in that some of the user's goals have been omitted, the generation and display of the plans may prompt the user to recognize the missing goals ("this plan is no good, because..."). These goals can then be inserted in the network.

Display of the causes of plan failure,

When one or more of the top-level goals cannot be achieved, the user is able to examine an explanation of the failure to achieve these goals, and is encouraged to amend the problem description so as to avoid goal conflict.

The Editing/Planning Cycle.

The user may continue the process of editing and replanning until satisfactory plans have been produced.

Applications.

The system has been applied to top-level planning for individuals, involving problems such as the choice of type of employment (self-employed vs. employed by an organization), and the choice of where to live (inner-city vs. suburbs vs. rural areas). These problems typically involve a number of top-level goals and values (e.g. autonomy vs. job security, convenience vs. aesthetics). Solution of the problems often requires the assignment of relative importance values to the various goals and values (prioritization). Use of the system was found to be a valuable means of clarifying the issues facing the individual concerned.

Problems encountered in the application of the system have centered upon knowledge elicitation. It is difficult for the user to express his/her goals and values in a form which is sufficiently structured to permit machine interpretation; conversely, it is difficult to implement an interpreter for free-form natural language propositions, because of the ambiguity of such propositions.

An earlier version of the system permitted free-form propositions; knowledge elicitation was straightforward, but the system was limited in its interpretation of this knowledge. Only the external attributes (e.g. importance, desirability) and relationships (e.g. implication) of propositions were taken into account.

The current version imposes restrictions upon the format of propositions; knowledge elicitation is more difficult, but the system is now able to interpret the content of propositions as well as their external attributes and relationships. For example, it is possible to detect contradictions involving different values for a single-valued attribute.

Experience with the present system has shown that there is a trade-off between the ease of Knowledge Elicitation, and the system's ability to interpret the knowledge elicited. This dilemma could be bypassed to some extent by the use of a natural language front-end.

In order to store the elicited propositions as logical formulae, the user's input would be parsed. Conversely, in order to display propositions to the user in a more intelligible form, the logical formulae would be converted back into English sentences.

There are reasons for doubting the value of such an interface. Experience with Expert Systems has shown that the process of reformulating knowledge in terms of more structured formalisms has helped experts to clarify their knowledge (e.g. Hayes-Roth et al, 1983). Clarification of strategic knowledge is one of the objectives of the present system. To allow knowledge to remain (from the user's viewpoint) in the form of vague and ambiguous natural-language statements might prove to be counter-productive.

Relation to Other Work.

The system bears a superficial resemblance to Shaw's computer implementations in the field of Personal Construct Theory (1980), and to Boose's Expertise Transfer System (1985) in the field of Knowledge Engineering.

Comparison with Shaw's work.

Both the present system and Shaw's various programs emphasise the interactive elicitation of personal models of the world. The present system differs from Shaw's in the following respects:

a. it is more complete, in that it includes Planning and Explanation modules. Shaw's programs only deal with the elicitation of a problem representation; the present system also generates, and explains, solutions to the problem;

b. it has a richer knowledge representation; Shaw's systems can only represent unary predicates; the present system can represent binary predicates as well;

c. it uses the "laddering" technique (Hinkle, 1965) for elicitation; Shaw's programs use the Repertory Grid technique;

d. it is less subtle in its treatment of the relationships between constructs; "fuzzy" implications are not represented;

e. it emphasises the use of the user's knowledge in the planning process, rather than the elaboration of the user's knowledge as an end in itself.

Comparison with Boose's ETS.

A similar comparison can be made with Boose's ETS, in that both are highly interactive, and both make use of the laddering technique for Knowledge Elicitation. The present design differs from ETS in the following ways:

a. it is designed to handle planning problems, whereas ETS is designed for analysis problems;

b. it is targeted at long-term, high-level, strategic problems; ETS is targeted at short-term, low-level, tactical problems.

General Comparison with other IKBS Systems.

Above all, what distinguishes the present system is its role in the elicitation and clarification of strategic knowledge. Most other systems are intended to support lower-level processes (e.g. scheduling, financial planning, project management) in which the strategic goals, values, priorities and assumptions have already been determined and are therefore "untouchable".

Conclusions.

The present system demonstrates the suitability of current IKBS techniques for Strategic Planning.

Since strategic processes are more critical than tactical processes, but involve fewer technical difficulties, research in this area is particularly well-motivated.

Extension and specialization of the present system for Corporate applications will result in the development of a Strategic Planner's Assistant.

Appendix 1: The Planning Procedures in Action.

The following example shows planning with a very small goal network. Nevertheless, it demonstrates some of the mechanisms involved. The system is capable of using much larger networks, even in its present PC-based implementation.

Comments are indicated by asterisks.

```
Entering the Planner.
Do you want to see each of the planning steps?
>>yes

Press the space bar after each line.

The goals to be achieved are, in order of importance:

  (I AM SOLVENT)  importance = 8
  (I AM INDEPENDENT)  importance = 7

* select a top-level goal:
(I TRIED TO ACHIEVE (I AM SOLVENT))

* retrieve a subordinate:
((I AM SOLVENT) WHEN:)
    (((I AM EMPLOYED))  )

* evaluate the benefits of the subordinate:
((I AM EMPLOYED) SUPPORTED THE FOLLOWING GOALS )
    (((I AM SOLVENT) IMPORTANCE = 8))
(THE NET BENEFIT OF ACHIEVING (I AM EMPLOYED) WAS 8)

* select a subgoal:
(I TRIED TO ACHIEVE (I AM EMPLOYED))

* retrieve a subordinate of the subgoal:
((I AM EMPLOYED) WHEN:)
   (((MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED))  )

* evaluate the benefits of the subordinate:
((MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED) SUPPORTED THE FOLLOWING
GOALS )
   (((I AM SOLVENT) IMPORTANCE = 8))
(THE NET BENEFIT OF ACHIEVING (MY TYPE-OF-EMPLOYMENT = SELF-
EMPLOYED) WAS 8)
```

```
* retrieve another subordinate:
((I AM EMPLOYED) WHEN:)
   (((MY TYPE-OF-EMPLOYMENT = EMPLOYED-BY-ORGANIZATION))  )

* evaluate the benefits of the second subordinate:
((MY TYPE-OF-EMPLOYMENT = EMPLOYED-BY-ORGANIZATION) SUPPORTED THE
FOLLOWING GOALS )
   (((I AM SOLVENT) IMPORTANCE = 8))
((MY TYPE-OF-EMPLOYMENT = EMPLOYED-BY-ORGANIZATION) NEGATED THE
FOLLOWING GOALS )
   (((I AM INDEPENDENT) IMPORTANCE = 7))
(THE NET BENEFIT OF ACHIEVING (MY TYPE-OF-EMPLOYMENT = EMPLOYED-
BY-ORGANIZATION) WAS 1)

* select the most beneficial subgoal:
(I TRIED TO ACHIEVE (MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED))
((MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED) WAS A PRIMITIVE SUBGOAL)
((MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED) WAS ACHIEVED)

* propagate success to the parent goal:
((I AM EMPLOYED) WAS ACHIEVED)

* propagate success to the top-level goal:
((I AM SOLVENT) WAS ACHIEVED)

* select the next top-level goal:
(I TRIED TO ACHIEVE (I AM INDEPENDENT))

* retrieve a negative subordinate:
((I AM INDEPENDENT) CANNOT BE ACHIEVED WHEN ((MY TYPE-OF-
EMPLOYMENT = EMPLOYED-BY-ORGANIZATION)))

* evaluate the status of the subordinate:
((MY TYPE-OF-EMPLOYMENT = EMPLOYED-BY-ORGANIZATION) WAS ALREADY
NEGATED.)

* propagate success to the parent goal:
((I AM INDEPENDENT) WAS ACHIEVED)

Plan Results:

The following goals can be achieved:

  (I AM SOLVENT)  importance = 8
  (I AM INDEPENDENT)  importance = 7

None of the goals have failed.

The following preconditions are required:

  (MY TYPE-OF-EMPLOYMENT = SELF-EMPLOYED)
```

References.

Boose, J.H. A Knowledge Acquisition Program for Expert Systems based on Personal Construct Psychology. IJMMS, 23 (1985), 495-525.

Carbonell, J.G. POLITICS: Automated Ideological Reasoning. Cognitive Science, 2 (1978), 27-51.

Gaines, B. & Shaw, M.L.G. New Directions in the analysis and interactive elicitation of personal construct systems. International Journal of Man-Machine Studies, 13 (1980), 81-116.

Hayes-Roth, F., Waterman, D.A., & Lenat, D.B. Building Expert Systems. Addison-Wesley, 1983.

Hinkle, D.N. The Change of Personal Constructs from the Viewpoint of a Theory of Implications. PhD Thesis, Ohio State University, 1965.

Shaw, M.L.G. On Becoming a Personal Scientist: Interactive Computer Elicitation of Personal Models of the World. Academic Press, 1980.

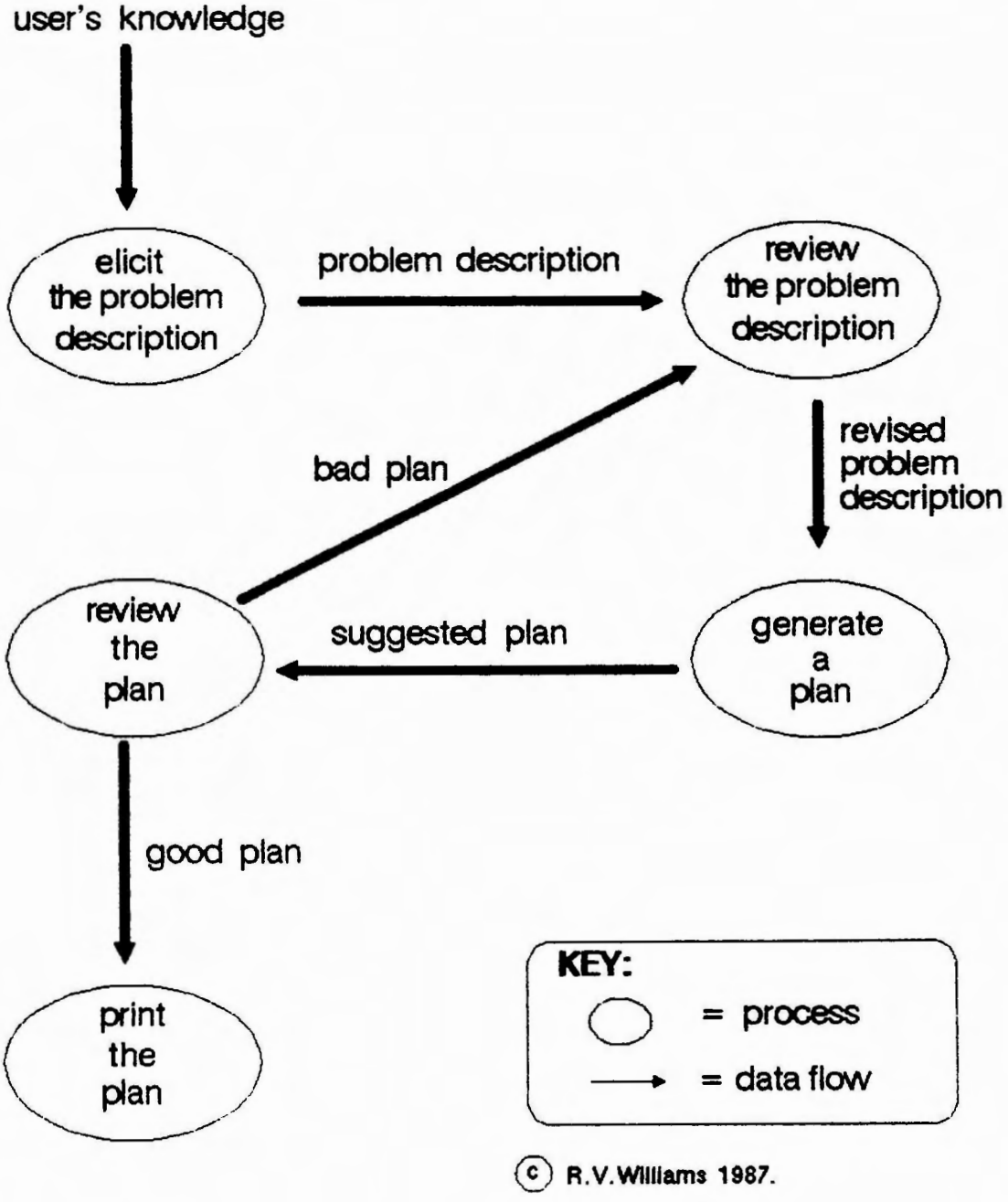
Wilensky, R. Planning and Understanding: A Computational Approach to Human Reasoning. Addison-Wesley, 1983.

Williams, R.V. Reasoning about Goals and Values in Strategic Planning. MSc Dissertation, University of Warwick, 1986.

Diagram 1

# Interactive Planning System: Overview of Processes

Date: 6/6/87
Author: Richard Williams




© R.V.Williams 1987.

# EXPLANATION BY CONTEXT RECOGNITION

**S. Thorpe**

Inference & Learning Group,
LRI, Universite Paris-Sud & CNRS
Batiment 490-91405,
Orsay, FRANCE.

ABSTRACT

The automatic recognition of students' misconceptions should become feasible when common-sense knowledge is encoded. Education's traditional remedy for a student's wrong answer has been merely to supply the right one. But our system would enable the student to comment on material presented to him, and if his comment indicates a disagreement with the teacher, it would ask him a question to find out the cause of the disagreement. The student's and teacher's explanations for a given fact are represented in terms of data-base "contexts" from which the fact is derivable. The method uses de Kleer's Assumption-based Truth Maintenance Systems [1, 2].

## 1 - Modelling

ICAI presupposes an intelligent dialog with the student. The GUIDON system derived from MYCIN searches for weak points in a student's knowledge and "entraps" him with questions about them [9]. Many writers agree that ICAI systems should consist of three modules: one to model the student, one to model the domain and one incorporating the pedagogical skills of the teacher [3, 4, 5]. Some ICAI systems do in fact contain three such modules. But under our proposal, both the teacher and the student would be modelled by using a truth maintenance system which enables conflicting beliefs to be processed simultaneously. Our representation is based on PROLOG.

### 1.1 - Modelling the lesson

We model both the student and the teacher as consisting of a "readily-accessible memory" and a "main memory". The main memory contains all the knowledge that they had before the lesson, while the ready memory contains their immediate understanding of the facts being presented in the lesson. We also refer to the memories representing the teacher as the "ideal" memories. Our suggestions about the boundary between ready memory and main memory are inevitably tentative, since we cannot guess at the extent to which the feasibility limits on memory will have changed by the time the necessary common-sense knowledge can be encoded.

Teaching in the classroom is modelled as an interaction between the student's ready memory and the teacher's or ideal memory. The student is offered the opportunity to comment on each sentence presented to him, and these comments are represented as the contents of his ready memory. We assume that he is always self-consistent, i.e. that there is no contradiction between his main memory and his ready memory. We also assume that he and the teacher can communicate, which requires their main memories not to conflict too violently.

### 1.2 - Nature of the Search

Each sentence presented to the student is a datum which could be explained through common-sense knowledge in many different ways. We expect the student's comment to be a tentative explanation of the study item presented to him, so that, taken in conjunction with his other beliefs, his comment is likely to entail it. To find out what these other beliefs are, the system will have to build some sort of bridge between the item presented and the student's comment. Many different rules of common-sense will be applicable, and there must also be rules specific to the domain concerned (history in the case of our example below). As described below, the student's beliefs are found by propagating the study item and his comment on it through these rules. However, ways must be found of giving precedence to rules which seem more relevant to the his beliefs.

### 1.3 - The Assumption-based Truth Maintenance System

If we want to model the student right from the first datum presented to him, the first few responses he gives us will leave many blank spaces in our model of his beliefs. A serious weakness of previous modelling systems has been their need to collect a sizeable number of responses from the student before they could begin to model him. We propose to keep track of every alternative way of filling in the blank spaces in the model. But the means of examining the alternatives right from the start have been provided by new ways of managing dependencies in problem-solving [1, 2, 7].

Suppose, for example, that we are querying whether Donald walks on two legs:

:- biped(Donald)

The first rule suggests deriving this from the fact that he is a man:

biped(x) :- man(x)

Thus we are led to ask whether Donald is a man.

:- man(Donald)

However, another common-sense rule could also explain why Donald walks on two legs: it may be because he is a duck. So corresponding to our first query-path, we have the following:

```
:- biped(Donald)
biped(x) :- duck(x)
:- duck(Donald)
```

But common sense also tells us that nothing can be both a man and a duck:

```
NOT-man(x) :-  duck(x)
NOT-duck(x) :-  man(x)
```

We give details on our use of NOT in section 2.3. If we take a rule and a subquery from above, we have that Donald is not a duck:

```
:- man(Donald)
NOT-duck(x) :-  man(x)
```

But from this and the other subquery:

```
:- duck(Donald)
```

it is impossible to derive the empty clause. This means that our basis is inconsistent. Both our explanations are potentially right, so the problem is how to keep working on both of them at the same time without contradicting ourselves. An Assumption-based Truth Maintenance System [1, 2] is able to do this because it marks each datum with a label describing the contexts under which it holds. A context for a datum includes the whole chain of rules and facts consituting its derivation. It is the closure under deduction of a consistent set of assumptions which entail it. This marking of the data enables them to co-exist in a single state as follows:

```
        CONTEXT 1                          CONTEXT 2
:- biped(Donald)                   :- biped(Donald)
biped(x) :- duck(x)                biped(x) :- man(x)
:- duck(Donald)                    :- man(Donald)
NOT-duck(x) :-  man(x)             NOT-man(x) :-  duck(x)
```

The incompatibility between ducks and men causes no problems here. CONTEXT 1, by querying and hence denying that Donald is a duck, affirms that he is a man, while CONTEXT 2, by querying and hence denying that Donald is a man, affirms that he is a duck.

In this way, the ATMS performs its task of keeping track of belief states. For each contradiction encountered, it marks the set of assumptions as inconsistent and updates the labels describing the contexts of the data. It does this efficiently because it can infer the new contexts for each node from its old label instead of inferring them directly from the assertions. In the above example, because the ATMS has marked the union of CONTEXT 1 and CONTEXT 2 inconsistent, its label-checking enables it to avoid deriving anything from their union or the supersets of their union.

## 2 - Interaction with the student

The student is presented with study data and each time he has an opportunity to start a dialog by typing in a comment. In order to assess this, we input it as a query to the ideal memory.

Three possible kinds of case are envisioned for the response:

- In the first case, the ideal memory will give the response "YES" to the the student's comment.
- In the second case, it will give the response "NO". This will happen if his comment contradicts knowledge in ideal main memory.
- The third case is the most interesting. Here, the system's response is neither "YES" nor "NO", but a diagnostic question. This arises when something seems to be lacking in the student's comment. By propagating the student's answer through ideal memory, the system is then able to identify what we shall call the "cause" of his misconception.

Here are examples for each of the three cases. They are converted into logical clauses which can be retrieved automatically by a PROLOG interpreter. A realistic implementation would of course require a natural- language front-end to handle the student's input, as well as the context recognition mechanism described in this paper. We shall work on this, and are keeping in touch with work on another system which already uses natural language input for a similar purpose [8]. It recognizes pre-stocked misconceptions while teaching psychology.

### 2.1 - The datum presented to the student

As each item of his study material is presented to him, the student has the option of commenting on it. Suppose that the lesson is about seventeenth-century France, and that the study datum currently under review is:

```
D1:              Louis XIV wore a  wig.
```

Converting this into a query, suppose that the front-end decides to represent it as:

:- wig-of(wig1, Louis-XIV)

Let us assume here that all the rules whose consequents match with the student's comment will now be activated, since one of them should lead to the student's explanation of why Louis XIV wore a wig. In reality, we intend to use micro-worlds to limit the search space, as indicated in section 3.

One of the rules available to explain the wearing of wigs is the following historical one:

```
R1:
FORALL x,  EXISTS y,  [[man(x) & noble(x) & lives-in-the-17th-
century(x)] -->  wig-of(y, (ornament(y)), x)]
```

For the sake of clear presentation, we postpone treatment of the Skolem function (ornament(y)) until section 3, and convert only the other part of R1 into PROLOG form:

```
wig-of(y, x) :- man(x) & noble(x) & lives-in-the-17th-century(x)
```

The following historical facts will be in the data base:

```
man(Louis-XIV) :-
noble(Louis-XIV) :-
lives-in-the-17th-century(Louis-XIV) :-
```

So the query derived from D1 above matches with the head of the clause derived from R1, giving rise to three further queries which match with the facts in the data-base. This results in confirmation through the empty clause.

### 2.2 - First case: Student's comment is correct

The teacher's preferred explanation for a datum is represented by the context for the datum whose contents are all already included in ideal memory. (This context could also be associated with the datum in advance.) The student's comment on D1 will be answered "YES" if it parses to a reasonable subset of the facts inferrable in this way from D1, for example, if it is like one of the following:

```
D'1:    Wigs were a normal part of what
        seventeenth-century noblemen wore.

D''1:   Louis XIV wore a wig as an ornament.
```

The system would represent these in the form of complex queries such as the following:

```
        Q'1:
:- lives-in-the-17th-century(Louis-XIV)
:- noble(Louis-XIV)

        Q''1:
:- wig-of(wig1,  Louis-XIV)
:- ornament(wig1)
```

Section 2.1 above indicates how the first three elements of Q'1 and Q"1 would be handled. The treatment of the query-element

```
:- ornament(wig1)
```

is accounted for in Section 3. Either Q'1 or Q"1 would get the response "YES".

**2.3 - Second case: The comment contradicts knowledge**

We have to allow for the possibility of the student giving a totally mistaken comment, such as:

```
D'2:        Bald women are ridiculed, so Louis XIV
            kept her baldness secret by wearing a wig.
```

This would be parsed to a query containing such clauses as the following:

```
        Q2:
:- wig-of(wig1, Louis-XIV)
:- woman(Louis-XIV)
:- saves-from-ridicule(wig1, Louis-XIV)
```

When the query-element :- woman(Louis-XIV) is invoked, it combines with the rule:

```
NOT-man(x)  :- woman(x)
```

to leave us with the assertion:

```
NOT-man(Louis-XIV)  :-
```

which can be recognized as a direct negation of the fact:

```
man(Louis-XIV)  :-
```

Following an intuitive line of thought, we wish to leave aside the Closed World Assumption, and to stipulate that a negative answer will be given only if the system finds a direct negation. So in this case, it tells the student "NO".

It may be felt that our abandonment of the Closed World Assumption is Utopian, but this is not the case. A superset of PROLOG containing two distinct NOTs has been developed in our research team by A. Marquez [unpublished results].

NOT1 is the classical NOT of PROLOG, which assumes a closed world, i.e. it will answer "NO" to the query A? when it fails to prove A. NOT2 is the NOT of classical logic. It answers "NO" to

the query A? only when it can disprove A. In practice, the use of NOT2 amounts to having a full First Order Logic theorem prover. In order to avoid the complexity of First Order Logic proofs, Marquez' system requires its user to provide a partial ordering of predicates when it uses NOT2.

### 2.4 - Third case: The misconception is recognized

Recall that in our account of the presentation of the study datum to the student in section 2.1 we required the activation of all the rules whose heads match the query:

```
:- wig-of(wig1, Louis-XIV)
```

Another of the rules available to explain the wearing of wigs is the following common-sense one:

```
R2:
FORALL x,  EXISTS y,     [[human(x) & disguises-himself(x)]
                  -->  wig-of(y, (disguise(y)), x)]]
```

The half of this which interests us now will be converted into PROLOG form as follows:

```
wig-of(y, x) :- human(x) & disguises-himself(x)
```

The first element of the tail will match a fact in the data base:

```
human(Louis-XIV) :-
```

However, no teacher is likely to know whether or not Louis XIV disguised himself. Hence there will be no fact and no direct negation in our database corresponding to the second element. So, contrary to the Closed World Assumption, the assertion disguises-himself(Louis-XIV) is marked as unresolved.

Suppose now that the student's comment is:

```
D'3:          Louis XIV wanted to have fun.
```

This can be translated to the query:

```
:- has-fun(Louis-XIV)
```

and will activate rules that give all the different ways of having fun. Among them will be the following:

```
has-fun(x) :- human(x) & disguises-himself(x)
```

The first sub-query here will be confirmed by the fact

```
human(Louis-XIV) :-
```

But there will be no fact and no direct negation in our database corresponding to the sub-query on disguises-himself(Louis-XIV). So this assertion is again marked as unresolved.

The available data must now be put together. As noted in section 2.1, the query on the datum presented to the student yielded a mass of potential explanations of why Louis XIV wore a wig. Each of

them will have been left incomplete when it failed to match with an appropriate fact about Louis XIV. Each will have resulted in a sub-query being marked as unresolved. Among them was the explanation derived through the common- sense rule R2, which also reached an impasse over the sub-query:

```
disguises-himself(Louis-XIV)?.
```

But at that stage there was no reason to suppose that any of these explanations of wig-wearing was believed by the student.

When the data-base was queried on the student's comment, the result was a mass of potential explanations of Louis XIV having fun, but without any indication of their relevance to his wig. However, there is an element which both sets of explanations have in common, namely, the fact that they both marked the derived query:

```
:- disguises-himself(Louis-XIV)
```

as unresolved. Thus, an intersection test indicates that the notion of disguise could be the explanation of wig-wearing which the student has in mind when he comments that Louis XIV was having fun.

In order to decide whether it is the explanation, the system must ask the student:

```
D*      Do you think Louis XIV was disguising himself?
```

This is the diagnostic question mentioned above in section 2. If the student replies "YES", then disguises-himself(Louis-XIV) is added as a fact to the context representing the student's beliefs.

### 3 - Micro-worlds

The next step should be to provoke a discussion. Where the student's context is recognized and made explicit, it can be assessed in comparison with the context preferred by the teacher, and further relevant material may be presented. This second stage of our work is still in a preliminary phase, but we intend to use micro-worlds to facilitate access to the relevant knowledge. A microworld is a portion of the database containing rules and facts about a certain aspect of knowledge, which is activated through a recognition function.

In section 2.1 we gave a clause representation of a common-sense rule reprinted here for convenience:

```
R1:
FORALL x,  EXISTS y,     [[man(x) & noble(x) & lives-in-the-17th-century(x)]
                  -->  wig-of(y, (ornament(y)), x)]
```

We represented R1 by a clause whose head was wig-of(y, x) but postponed representing its Skolem function (ornament(y)). This will require a second clause as follows:

```
ornament(y) :- man(x) & noble(x) & lives-in-the-17th-century(x)
```

This clause represents the teacher's understanding of the wig, which he sees as an ornament. A similar problem arises for the rule given in section 2.4:

```
R2:
FORALL x,  EXISTS y,     [[human(x) & disguises-himself(x)]
                    -->  wig-of(y, (disguise(y)), x)]]
```

This needs to be represented not only by the clause whose head is wig-of(y, x), but also by the following:

```
disguise(y) :- human(x) & disguises-himself(x)
```

This clause represents the student's understanding of the wig, which he sees as a disguise. In both the clauses representing Skolem functions, the variable in the head differs from the others, and they need to be called in a special way, after the activation of the wig-of(y, x) clauses associated with them. We envisage that after the facts in these clauses have been confirmed or marked as unresolved, the system will chain forward through the recognition functions to identify the relevant knowledge.

If the student's reply to the diagnostic question D* is "NO", the negation of D* is added to his ready memory, and the search for an explanation of his belief may continue with the activation of other micro-worlds.

The difficulty with the system is that it requires an enormous collection of commonsense knowledge, similar in principle to the CYC project [6]. The reward is that it could correct each student's misconceptions individually.

**Acknowledgments**

We would like to thank Gheorghe Tecuci for helpful advice. The "wig example" is due to S. Duchenoy [4].

**References**

[1] J. de Kleer, Choices without Backtracking, Proceedings of the AAAI-84 Conference, (1984) 79-85

[2] J. de Kleer, An Assumption-based Truth Maintenance System, Artificial Intelligence 28, 1986 127-162.

[3] A. Barr & E. Feigenbaum: The Handbook of AI, vol.2 (Pitman 1982) 221 - 294.

[4] Costa E., Duchenoy S., Kodratoff Y., A resolution based method for discovering students' misconceptions, J. Self (ed), Intelligent Computer- Assisted Instruction (Chapman & Hall, 1987)

[5] T. O'Shea & J. Self, Learning and Teaching with Computers: AI in Education (Harvester Press, Brighton, 1983)

[6] D. Lenat, M. Prakash & M. Shepherd, CYC: Using Common-sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks, AI Magazine vol. 6, no. 4 (1986) 65 - 85.

[7] J.P. Martins and S.C. Shapiro, Reasoning in multiple belief spaces, Proceedings of the 8th IJCAI, Karlsruhe, 1983.

[8] J.-F. Le Ny, L. Carite & S. Poitrenaud, Construction of individualized texts for the transmission of knowledge through discourse, in I.Kurcz, G.W. Shugar & D.H. Danks (editors): Knowledge and Language, Elsevier Science Publishers B.V. (North-Holland)

[9] B.G. Buchanan & E.H. Shortliffe, Rule-based Expert Systems (Addison-Wesley 1985), pp. 483 ff.

# AN EXPERIMENTAL PROTOCOL FOR THE ACQUISITION OF EXAMPLES FOR LEARNING

Jim Blythe, David Needham

GEC Research, West Hanningfield Rd., Chelmsford CM2 8HN, England.

Patrick Corsi

Cognitech, 167 Rue de Chevaleret, 75013 Paris, France.

**KEY-WORDS:**
machine learning, noisy data, example-base creation, user models, acquisition methodology, questionnaire tool, human factors, plant pathology.

**ABSTRACT**

The training examples used to learn a rule base inductively are of central importance to the quality of the knowledge learned. In order to produce a learning system that can be used to construct knowledge bases in a variety of domains, it is important to provide a protocol for the acquisition of examples that can be applied to these domains. In this paper, we present a case study of example acquisition in the field of plant pathology. We first note the types of noise or inaccuracy that may be present in the domain, then consider a simple model of the acquisition process. The examples can be usefully described by non-experts, reducing the cost of knowledge acquisition and combatting one type of noise.

We generalise the experience of the case study to suggest key points for a protocol for knowledge acquisition in machine learning. This takes various aspects of noise into account, and allows the final representation language to be oriented towards the targeted users of the system, as well as the experts.

## 1. Introduction

The INSTIL project aims to build a learning system adapted for use in real-life environments. To that end, the system should be developed in conjunction with a domain of expertise. The domain of plant disease diagnosis was chosen at the beginning of the project because of two main observations:

- Some machine learning and KBS implementations already existed in this field at the time (eg Soja [Michalski & Chilausky 81]) and more were on the way (eg Wheat Counselor by ICI), and
- an expert system on tomato plant pathology had already been developed by INRA (Institut National de Recherche Agronomique) and Cognitech, thus allowing the project to be evaluated by direct comparison with an existing system at the conclusion of the project.

One problem that is still to be overcome in applying learning systems in such a domain is that of noise, which we take to mean incorrect or incomplete data in the training examples. Dealing with noise is the major work of the INSTIL project. The approach taken is based on the integration of three existing learning systems, all of which have been developed and used by the partners, and which have different strengths and weaknesses [Smallman 85]. From them, we can produce a system for rule induction that combines a powerful representation with low sensitivity to noise.

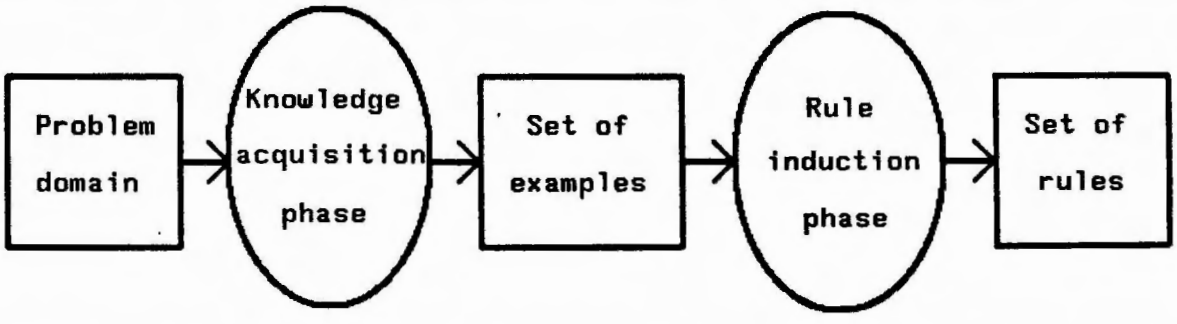



Figure 1: the complete process of constructing a knowledge base by machine learning.

However, we do not concentrate solely on the rule induction phase for treating noise in the process of learning. While a great deal of effort has recently been put in to the problems that noise causes for the machine learning task [Quinlan 86], [Bratko 87], [Niblett 87], most of this work has focussed on the rule construction phase, concentrating on making it less sensitive to individual examples that may be erroneous. Principal techniques involved in this task have been pruning [Briemann et al. 84] or truncation [Mozetic 86] of the rule base to make it less sensitive to

detail.

We view the treatment of noise as a process that should affect every stage of the learning process, as shown in figure 1. A particularly important phase is that of example acquisition, which is the subject of this paper. We present the methodology that the project has developed for conducting knowledge acquisition in a noisy domain, and provide a case study in the domain of tomato pathology. This phase took place between April and June 1987.

The idea of treating noise at the acquisition stage is *not* based on eliminating all noise from the examples: this would undermine the capability of the induction module to respond to the noise that is present in the domain both at consult time and learn time [Quinlan 86]. Our study aims rather at classifying the noise that will be present by source and effect, to maximise the information available to the learning system, and at reducing the noise that is only present during the acquisition phase.

### 1.1. Rationale for the acquisition of training examples

In the next section we develop some of the background necessary to construct a simple model of the process of example acquisition. Central to this work is the desire to produce guidelines for gathering examples that can help to combat noise in a domain that is subject to it. Such a protocol will be necessary to ensure that the final system is well applicable to a range of problem domains. It is also essential in order to deal with noise in a coordinated way. To do this, we investigate the types of noise that can particularly affect the process of knowledge acquisition, and develop a simple model of the process.

One manifestation of the noise inherent in a domain is that *different people would describe the same event differently*, even if they use the same description language. For example, the threshold at which red becomes orange is not a universal standard, nor is it clear when a leaf's edge stops being smooth to become jagged. This phenomenon seems to be strongly linked with the 'skill' level of the observer, since an expert can see much more detail in his domain than a novice.

This observation leads us to a major principle of our approach to knowledge acquisition: *that examples should be described by a number of people, preferably of the same level of skill in the domain as the targeted end user*. This is in contrast to the classical expert systems approach, where the skilled knowledge engineer derives the system from a few experts.

The work of *diagnosis* must of course be done by an expert, in order to capture his or her knowledge. Thus the scene is set for a number of active agents of differing skills and tasks in the knowledge acquisition phase. We next analyse the degree to which the different types of noise identified affect each agent.

Finally, we take into account the different types and sources of examples that may be used for rule induction. It is estimated that it would take seven years to see an example of every type of tomato disease that can occur in France. Thus a reasonable system must make use of examples taken from other sources, e.g. photographs and the memories of experts. These must be treated in different ways as described below, although this is mainly a problem for the learning system, and is outside the scope of this paper.

In section 3, we present some of our experiences of knowledge acquisition in the case study of the tomato plant domain. This is compared with the ideas developed in section 2. In the final section, we present partial results of the study, and discuss how the lessons learned might be generalised to apply to other domains.

## 2. THE ACQUISITION METHODOLOGY

### 2.1. Types of noise to be treated

We now come to look more closely at the kinds of noise that are present during the task. We are not concerned in this paper with their treatment after the acquisition phase is completed, or with other types of noise that may then arise. The reader is referred to [Kodratoff et al. 86] for the previous work on which this section is based. The consortium pinpointed 11 types of noise that have an impact on this phase, which are as follows. The number of + signs indicates increasing importance.

N1. A faulty description language:

Frequently a learning system fails because the description language for examples has not been rich enough to express the differences between them. The use of a pilot phase (see below), and the analysis of some examples by an expert helps to alleviate this problem.

N2. Descriptors that are costly to check: +

Very often, descriptors are costly to check since:

(a) they necessitate killing the plant or require a lot of

plants, or

(b) when a sample is not complete (consists only of parts of plants) it would cost time to obtain certain measurements.

N3. Descriptors hard to notice: +++

Very often, these descriptors are the most pertinent ones. They are also the ones the experts like to use for showing off (!) The eyes of the experts are accustomed to finding them, but not those of a technician; the expert can see signs that are hardly visible, and can recognise signs under a wide range of polymorphy.

N4. High level concepts similar and/or blurry: ++

Some diseases express similar symptoms under particular conditions. The expert himself is sometimes obliged to make laboratory tests. In fact, descriptors needed to differentiate such concepts are often costly or hard to notice.

N5. Randomness of natural phenomena: ++

Few natural domains are without some variation of features across the sample space. Any acquisition method will be at the mercy of factors beyond our control - this is one reason to make the operation as large as possible.

N6. Variation of tolerance of tests: +

May happen, particularly with accuracy of measurements (length of a plant).

N7. Mistake in providing a value: +

This will certainly be the case with non-naive users or plant specialists who may allow their views to be influenced by what they expect to see. As origins, we can mention (a) lack of attention, and (b) prejudiced view. When naive users make mistakes, they can often be corrected in the early stages of the experiments. The risk is decreased by on-line help and the spelling checker provided by an interactive questionnaire.

N8. Leaving out a descriptor: +++

This is inherent in the domain, because an example is often

incomplete. Beyond noting that a descriptor cannot be evaluated, we feel it is not appropriate to treat this type of noise during the acquisition phase, since choosing 'good' examples will lead to a bias that is not present when the system is in use. This problem is handled during the learning phase, where the system chooses the more 'reliable' descriptors.

N9. Wrong class attributed to an example: ++

This case probably has the same origin as N7, that is, prejudiced view.

N10. Bad examples with too many symptoms: ++

That could happen because normal (not diseased) plants present imperfections that could be described by a non-specialist of the plant. It might happen also in the multi-disease situation where one (or more) diseases are not declared.

N11. Bad or incomplete example space: +++

A law of the domain. In general, a set of training examples that span the problem domain may take several years to produce. This fact of plant pathology makes it very likely that, in the long run, a learning apprentice system capable, of adapting while being used, will be appropriate for the domain [Mitchell et al. 85].

In the next sections we develop the possible solutions we propose to minimise the effects of these types of noise. We stress again that treatment *must* occur throughout the learning process, and some types are more easily treated during the induction phase.

## 2.2. A typology of the domain actors.

This section is based both on the ideas mentioned in section 1, and on our practical experience. We differentiate three categories of personnel involved at some specific points during the search and collection of the examples. They are:

- The *Farmer* (F) is in charge of growing and curing the tomato plants. He often asks the Technician (see below) for help.

- The *Technician* (T) is knowledgeable about the tomato field, but

sometimes has to ask the expert's help for a difficult diagnosis. We should distinguish here between the "naive" technician (NT), who is an occasional user but not trained or accustomed to the questionnaire, and the "non-naive" technician (NNT) who knows the questionnaire well in some form or another.

- The *Expert* (E) is the agreed reference for expertise in the application field. He is the one whose task is to validate the results at each step of the data flow. His diagnoses shall always be kept as a major reference as far as all future performance and validation tests are concerned.

Types of noise have an impact on each category of user in the ways described below.

◊ The technician: N3, N7, N8 and N10. These probably depend on the technician's level of knowledge about the tomato. We may distinguish between the two types of technician:

- Naive Technician: *makes mistakes in answers*
  N3: he is not familiar with the descriptors in the questionnaire
  N7: he will probably generate less of this noise, because of the great attention be must pay to fill the questionnaire.

- Non-naive Technician: *presents bias about knowledge*
  N3: he will probably generate less of this, because he is used to thinking in terms of our descriptors.

◊ The Expert: *gives incomplete description of the world*
N9: the source of noise we expect is the incomplete diagnosis.
N4: probably comes from N5 or N2, or from a lack of laboratory tests.

## 2.3. Different types of 'example' to be used in a learning system.

There are two major types of examples that will be collected and used in a system - 'real' ones, that come from the field and are diagnosed by the expert, and 'synthetic' ones that must be used to supplement the base of training examples to provide an adequate coverage of information.

There are three subtypes in the category of real examples:

- Those cases which would normally come from the field to Technical Centres to be diagnosed. The example is described after reception and before diagnosis by an expert. Their acquisition is completely passive.

- Those cases that come to be known to technicians and must be collected. This is a slightly more active form of acquisition.

- Those cases that would otherwise remain unknown to the technicians. By introducing our work force into the field, under the guidance of technicians and other personnel, an active search can bring results. This is the most time-consuming process.

A mixture of the passive and active methods of receiving examples should be used in such a domain as this if possible, to overcome any bias that may be introduced by using only one form of acquisition.

The 'synthetic' examples can be split into those taken from photographs, where certain descriptors often have to be guessed, those taken from written records of previous seasons, either on-line or not, and those that come from an expert's memory.

It is important to remember that the synthetic examples have in some sense been singled out and kept as "case studies", often used for teaching purposes. In general, they are either proto-typical members of their class or highly atypical, kept for their strangeness. These factors are important in the design of a learning system capable of accepting both types of example, but do not concern us here, except to note it is important to differentiate them.

| Items / Examples | Interest | Inconvenience | Noise |
|---|---|---|---|
| Natural examples | Actual farmers problem | - Non complete<br>- Weather dependant | N1 to N11 |
| Expert memory | Completeness | Idealised examples | N8, N10 |
| Descriptive papers | Case library | Other representation | N8, N2 |
| Photographs | Case library | Partial representation | N8, N3 |

Figure 2: the types of example.

## 3. A CASE STUDY IN THE TOMATO PLANT DOMAIN

As mentioned in section 1, the acquisition phase for our test domain of tomato plant pathology was completed in early Summer 1987. In France, INRA is widely recognised as the best organisation for conducting large-scale experiments in plant pathology. They provided the experts with whom Cognitech built the original TOM expert system, and also provide help in our current work.

Any individual case study of example acquisition will have slightly different circumstances, leading to a slightly different organisation of domain actors and a slightly different protocol. In this section, we describe the organisation of people in Avignon, and their interaction. In order to handle a large flow of examples, an automated questionnaire has been developed by Cognitech, providing a structured way to store examples, with help facilities about the description space. We discuss this system. We also look at how the acquisition phase can be controlled and calibrated using a pilot phase.

### 3.1. Choice of site

We intend the final system developed from this study to be used across France and possibly in other countries as well, so the examples gathered should ideally reflect the wide variation of growing conditions that exist in this country. However, it was found that the time taken to develop the necessary relations with farmers and experts in a given area was prohibitive to the establishment of a large number of sites around France. Within INRA, the area of Avignon is of prime importance for research on tomato pathology, because it is a region where the tomato is a major crop, and there is a large variation in conditions over a relatively small area. Since therefore many of INRA's experts and trained technicians were available at the centre in Avignon, it was chosen as our major area for example acquisition.

### 3.2. Domain actors

The classes of people active in Avignon are the three mentioned in section 2.2. At the start of the acquisition phase, there were good relations already established between INRA and the farmers of the area. We introduced some people into the area as technicians, to collect examples actively for us, and they worked through these established relations. We note the following specific problems in the current domain:

- There is some loss of information between the farmer and the expert, because of a typical reluctance to call an expert in. We therefore lose a few examples that way.

- There is the possibility of information loss due to the presence of various hierarchy levels and some old communications habits within the rural community, which the filler may not always be accustomed to. This loss he has to detect.

- finally, it seems better to describe (when possible) the tomato at the site where it grows, in order to avoid forgetting some aspects or if an unavailable piece of information comes to be needed.

### 3.3. The Questionnaire

The questionnaire in use in Avignon is a software tool for knowledge acquisition that structures and formats the examples as they are entered. It was developed on a SUN-3 and ported to a BULL DPS8/Multics system at INRA. Consultations of the questionnaire are made through TRANSPAC, the French public network.

The questionnaire contains a tree structure of menus, which match the hierarchy of objects in the domain knowledge. It allows the user to make comments about any descriptor, as well as enter the objects and values that are processed. When requested, it provides on-line help facilities and can give default values for descriptors. In order to help noise elimination at later stages, the questionnaire also makes a note of the date, time and user's identity.

### 3.4. The control of the acquisition phase

We found it useful to carry out the acquisition process for two weeks before we began to accept the results, in order to remove the possibility of noise introduced by lack of familiarity with the system. We view this period as the control of the acquisition phase, which may serve these purposes:

- *validation* of the collection of the examples.

- *calibration* of the parameters of noise (costs, defaults etc.), to estimate the noise produced during the acquisition phase.

- the establishment of links with workers in the field and experts.

- eventually, the trace of this event enables us to propose a methodology of data acquisition for an INSTIL environment to generate a complete database.

The examples produced during this period were examined both by experts and technicians. Likely areas of vulnerability to noise were brought to light, and in addition several improvements to the description language being used were made, where it had been found to be inadequate. We note here that this control period inevitably introduced some bias in the tomato domain, because it took place at the start of the season, and all the plants were young, but we do not see how this could have been avoided without spending an entire season as a control phase.

## 4. RESULTS AND CONCLUSIONS

In this section we provide the final results of the acquisition phase covering the pilot phase and another two and a half months of data collection. From our initial experience we make some suggestions about what a more general knowledge acquisition phase for an inductively generated knowledge based system might be like.

### 4.1. The diseases covered within the crop season.

During the acquisition phase we obtained 203 exploitable real examples and 105 exploitable synthetic examples from photographs. They together cover 46 different diseases, representing most of the known tomato plant diseases. Other examples were collected, but rated unexploitable for various reasons (communications failure, lack of descriptors, etc).

### 4.2. A proposed methodology for example acquisition

Some of our experiences with the test domain are individual to that domain, but many can be generalised to form a set of guidelines for the task of example acquisition. We now summarise the work presented here by way of considering these guidelines.

- Domain actors
  In most domains, the target users will have a different way of describing the domain from the experts. This can be a problem in classical expert systems. We propose that examples should be described by people of the same skill level as the end users, working in the "field" with those users. The examples should be classified by the expert, however.

- Documentation
  The examples gathered should be carefully documented, in case they are later questioned by a learning system. In particular, it is useful to save the date and filler's identity. This is best done by questionnaire software that can also provide help

about the description language.

- Pilot phase
  It will often be an advantage to spend a small percentage of the total time available in a 'dry run' or pilot phase. This allows the users to get familiar with the system, the experts to calibrate various types of noise that may be introduced, and provides an opportunity for the questionnaire to be modified if necessary. This phase should preferably involve some test runs with the learning system to examine the knowledge being formed.

This project has looked extensively at the possibility of using machine learning techniques to generate commercial expert systems. Most of the research work in this field uses examples bases that are easy to acquire, such as are provided by medical records [Quinlan 86], [Bratko et al. 87], [Michalski et al. 86]. We have investigated the case where such bases are not so readily available.

We find that the task of acquisition can, indeed should, be shared between a number of technicians, rather than a few experts. It seems a quicker and cheaper process than conventional knowledge elicitation for expert systems. We also see that the process of actively creating the example base allows us to introduce noise handling techniques. Finally, the presence of end-users in the acquisition process helps ensure that the domain language used in the final system is oriented as far as possible towards the end user as well as the expert. This finding has a parallel with other areas of knowledge acquisition, [Solvberg et al. 87], where it is found useful to engage people of different levels of expertise in the different stages of the knowledge acquisition task.

**Acknowledgements**

This work was conducted as part of INSTIL, ESPRIT project 1063. We thank our partners in this project, Michel Manago and his colleagues at the Laboratoire de Recherche en Informatique, Universite de Paris-Sud, who made important contributions to the work. We are grateful to the INRA institute for providing the necessary human and material resources for the collection of examples. Special thanks go to Christine Piaton and Noel Conruyt, students in agriculture, and to M. Dominique Blancard, expert in tomato plant pathology at INRA - Avignon.

**References**

Bratko et al. 87, Bratko, I., Mozetic, I., Lavrac, N., "Automatic synthesis and compression of cardiological knowledge", *Machine Intelligence*

*11*, Hayes, Michie & Richards eds., Oxford University Press,

Briemann et al. 84, Briemann, Friedman, Olshen & Stone, *Classification and Regression Trees*, Wadsworth Press, 1984.

Kodratoff et al. 86, Kodratoff, Y., Manago, M., Blythe, J., Smallman, C., Andro, T., "The Integration of Numeric and Symbolic Techniques in Learning", presented at the *AAAI Workshop on Knowledge Acquisition*, Baanf, Canada, 1986

Michalski & Chilausky 81, "Knowledge Acquisition by Encoding Expert Rules versus Computer Induction From Examples: A Case Study Involving Soybean Pathology", in *Fuzzy Reasoning and its applications*.

Michalski et al. 86, Michalski, R., Mozetic, I., Hong, J., Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments", Intelligent Systems Group, Dept. of Comp. Sci., University of Illinois.

Mitchell et al. 85, Mitchell, T., Mahadevan, S., Steinberg, L., "LEAP: A Learning Apprentice for VLSI Design", in Proceedings of IJCAI-85, Los Angeles, August 1985.

Niblett 87, "Constructing Decision Trees in Noisy Domains", in *Progress in Machine Learning*, I. Bratko & N. Lavrac eds., Sigma Press 1987.

Quinlan 86, "The Effect of Noise on Concept Learning", in *Machine Learning, volume 2*, Michalski, Carbonell & Mitchell, eds., Morgan Kauffman 1986.

Smallman 85, Technical Annexe for ESPRIT project 1063: *The Integration of Numeric and Symbolic Concepts in Learning*, 1985.

Solvberg et al. 87, Solvberg, I., Nordbo, I., Vestli, M., "Visualization of knowledge to improve the knowledge acquisition process", SINTEF-RUNIT, University of Trondheim, Norway.

# Automatic Induction of Rules from Examples: A Critical Analysis of the ID3 Family of Rule Induction Systems

Dr.M.A.Bramer
School of Computing and Information Technology
Thames Polytechnic
Wellington Street
London SE18 6PF
England

May 1987

**Abstract**

An approach which has been strongly advocated in recent years - particularly in Britain - for overcoming the knowledge acquisition 'bottleneck' in developing Expert Systems is the use of techniques of 'rule induction', by which rules are automatically derived from sets of examples. The ID3 algorithm developed by Ross Quinlan has been particularly influential and has formed the basis for a number of commercial products. This paper presents a critical review of ID3 and its main derivatives and identifies a number of potential weaknesses in the approach, as a stimulus to further research. An alternative semi-automated approach which makes additional use of domain experts' subject knowledge is outlined.

## 1. Introduction

The Expert Systems of today contain a few hundred (or in rare cases a few thousand) rules and perform relatively straightforward tasks.

The Expert Systems of tomorrow may need to contain many thousands (or even tens of thousands) of rules to perform the more complex tasks which are likely to be required. How will these rules be found?

The customary techniques of knowledge acquisition (introspection, interviewing, protocol analysis, analysis of critical events, etc.) all seem significantly limited and past experience, even with the most acclaimed Expert Systems is not encouraging.

To take only two examples from many in the literature, Buchanan et al. (1969) describing their experience with Dendral state:

> "....one of the greatest bottlenecks....has been eliciting and programming new pieces of information...the theory does not exist in any sort of comprehensive codified form."

and Shortliffe (1976) says of MYCIN:

> "The formulation of new decision rules is no straightforward matter. Physicians have not in general structured their decision processes and the expert...may have difficulty describing the steps in reasoning that he uses to make decisions".

The difficulties frequently encountered in extracting rules from experts are not particularly surprising, on reflection. The skill of an expert lies in performing a given task, not in explaining to others how it should be done. To this may be added the natural reluctance which many experts may well feel to pass on their knowledge to an outsider, especially one who may appear to be threatening the expert's own position in his or her organisation (remembering the slogan 'in the knowledge lies the power').

An alternative approach to knowledge acquisition which has become increasingly popular in recent years - not least because of the advocacy of Professor Donald Michie and his co-workers at the Turing Institute in Glasgow - is the so-called Rule Induction approach.

Essentially, the idea is to take a database of example cases and use a program to discover the underlying rules automatically.

It could be argued that for experts to supply significant case examples to an induction program is a natural means of knowledge transfer, far more so than explaining how a skilled task is performed, and the hope is that induction will provide a practical way round the problem of the knowledge acquisition 'bottleneck'.

The most influential work on inducing rules from examples in recent years - at least in Britain - has been the ID3 algorithm of Ross Quinlan.

This algorithm was based originally on the 'concept learning system' (CLS) of Earl Hunt (Hunt, Marin and Stone, 1966), but differs from it in a number of respects.

ID3 was first described in Quinlan (1979) and has served as the basis for a number of other induction systems, including some commercial software packages. Quinlan (1986) is a good review of this work.

This paper presents a critical review of ID3 and its main derivatives and identifies a number of potential weaknesses as a stimulus to further research. An alternative semi-automated approach which makes additional use of domain experts' subject knowledge is also outlined.

In the next section, the ID3 family of algorithms is introduced and examples given of its use. Some extensions to the basic algorithm are also briefly described. In Section 3, Ross Quinlan's original experiments with the Chess endgame King and Rook Versus King and Knight are analysed. Section 4 presents a critical review of ID3 and related algorithms. This is followed in Section 5 by a brief description of an alternative semi-automated approach which makes additional use of domain experts' subject knowledge. The final two sections comprise overall conclusions and references.

(It should be noted that alternative approaches to automatic rule induction, such as that described by Michalski and Chilausky (1980), are not considered in this paper.)

## 2. Induction of Rules from Examples

### 2.1 Top-down Induction of Decision Trees

The standard technique used by the ID3 family of algorithms to induce rules from a database of examples (whether complete or partial) is known as the Top-Down Induction of Decision Trees (TDIDT).

This method is only applicable when the examples in the database are of a particular - but quite common - form.

In the standard formulation of the algorithm, the database is known as the training set and each example in the database is known as an instance.

It is assumed that there is a universe of objects, each of which belongs to one of a set of mutually exclusive classes.

Each instance in the training set must be a description of an object, together with the class to which it belongs.

Each object is described by the values of a collection of its attributes (the same attributes being used for each instance), each attribute taking one of a set of discrete, mutually exclusive values.

It is customary to depict the training set by means of a two-dimensional table, with one row per instance, and each column holding the values of a particular attribute or the class to which the object belongs.

The aim is to develop classification rules in the form of a decision tree which enables the class to which any object in the training set belongs to be determined from the values of its attributes.

As an example, Table 1 below is a training set containing data collected about eight Thames Polytechnic students, showing for each one:

| | |
|---|---|
| eyecolour | (brown or blue) |
| sex | (male or female) |
| hairlength | (long, short or bald) |

and the 'class' to which the student belongs, in this case whether the student is taking a degree in computing or in the humanities.

(Note that in the general case there is no reason for there to be only two classes. The important point is that the classes are mutually exclusive.)

| eyecolour | sex | hairlength | class |
|---|---|---|---|
| brown | male | bald | humanities |
| blue | male | short | computing |
| blue | male | short | computing |
| brown | female | long | computing |
| brown | female | short | humanities |
| brown | male | bald | humanities |
| brown | female | short | humanities |
| brown | male | short | humanities |

Table 1

The data in this training set can be represented in the form of a decision tree in a number of ways, two of which are shown as Figures 1 and 2 below.

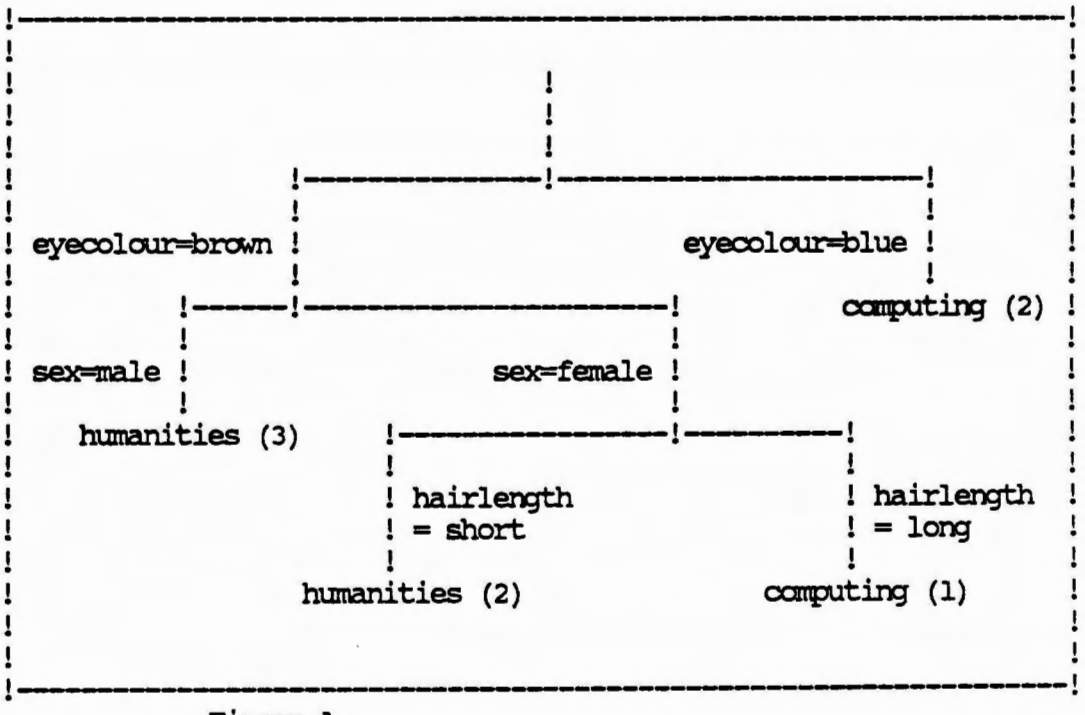



Figure 1

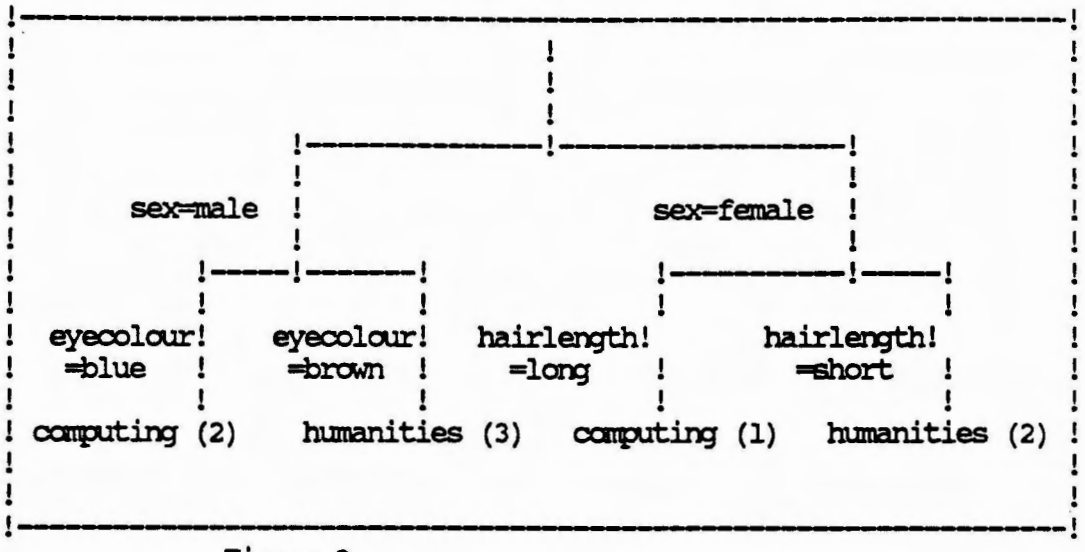



Figure 2

(The numbers in parentheses indicate the number of instances in the training set to which each branch corresponds.)

The basic induction algorithm used to produce decision trees, such as Figures 1 and 2 can be described informally as follows:

IF all the cases in the training set belong to the same class

THEN return the value of the class

ELSE (a) select an attribute A

(b) sort the training set into subsets each containing those instances with the same value for the selected attribute A

(c) return a tree with one branch for each subset, each branch having a descendant subtree or a class value produced by applying the algorithm recursively for each subset in turn.

(It is understood that the same attribute is never selected twice in the same branch and that values of A which lead to empty subsets at step (b) are discarded when returning the tree at step (c).)

Step (b) is known as splitting on attribute A.

The decision trees shown in Figures 1 and 2 were generated by making an arbitrary choice of the attribute to split on at step (a).

Provided the set of attributes is adequate, i.e. no two different instances have the same values of all the attributes but belong to different classes, any choice of attribute at step (a) will suffice to produce a (different) decision tree, providing of course that the same attribute is not chosen twice in the same branch. Thus, the algorithm is certain to terminate however the attributes are chosen.

Nevertheless, the choice of attribute is generally of great importance, as will be discussed in Sections 2.3 and 4.3 below.

It is also important to appreciate that although there will in general be many alternative possible choices of an adequate set of attributes to describe each instance, a good choice of attributes can greatly affect the practical usefulness of this method.

### 2.2 Decision Trees Viewed as Rules

It is normally helpful to regard such an induced decision tree as a set of rules, one corresponding to each branch of the tree. Working from left to right through the branches of Figure 1 gives the four rules:

IF eyecolour = brown AND sex = male THEN humanities
IF eyecolour = brown AND sex = female AND hairlength = short THEN humanities
IF eyecolour = brown AND sex = female AND hairlength = long THEN computing
IF eyecolour = blue THEN computing

(In passing, it should be noted that the branches in such a decision tree are not necessarily all of the same length.)

Naturally, the instances in the training set can themselves be written in the form of rules, one per instance, each having three components in its premise, e.g. the first instance can be written as:

IF eyecolour = brown AND sex = male AND hairlength = bald THEN humanities

The premises of the four rules corresponding to the decision tree in Figure 1 have an average of 2.25 components. These replace a training set comprising the equivalent of eight rules, each with three components in its premise.

Thus, even for this simple example and with an arbitrary choice of attribute to split on at each stage, it is significantly simpler to find the class to which any member of the training set belongs by using the induced set of rules than by examining the original data.

An alternative way of viewing an induced decision tree is therefore as a more compact means of storing the data in the training set with a considerable compression of data frequently being possible. This point will be discussed further in Section 4.2.

### 2.3 The ID3 Family of Rule Induction Algorithms

The ID3 algorithm of Ross Quinlan (Quinlan 1979, 1983) is an extension of the basic induction algorithm described in Section 2.1. ID3 aims at developing decision trees of the kind previously described, which are essentially equivalent to exact (i.e. non-probabilistic) rules. It was originally developed for domains in which no 'noise' is present in the data but Quinlan (1986) also describes experiments with noisy data.

An important point in the original design of ID3 was the need to construct decision trees for large (or very large) training sets,

where the basic induction algorithm described in Section 2.1 could not be used directly.

The approach adopted in the original version of ID3 was an iterative one based on the use of windows, i.e. relatively small subsets of the whole training set.

The basic ID3 algorithm is as follows:

(i) choose a subset of the whole training set to work with as an initial window

(ii) perform the basic induction algorithm (described in Section 2.1) on the instances in the window

(iii) find all cases where instances in the whole training set are wrongly classified by the rules induced by (ii)

(iv) select a replacement window (incorporating some of these misclassified instances)

(v) go back to step (ii)

The algorithm continues until there are no incorrectly classified instances at step (iii).

The algorithm as given here defines a family of rule-induction algorithms, depending on the method adopted to:

select the initial window at step (i)
select the attributes to split on when performing the basic induction algorithm at step (ii)
select the replacement window at step (iv).

Any algorithm of this basic type may be described as 'ID3-like'.

In Quinlan's original ID3 experiments (described in Section 4), the choice of the attributes to split on in the basic induction algorithm was based on the use of an ad hoc formula. This was subsequently replaced by a formula derived from information theory which essentially amounts to choosing the attribute which maximizes the expected gain of information from applying the additional test. Details are given in Quinlan (1986).

There are two alternative approaches to choosing the replacement window at step (iv) described in Quinlan (1979): the fixed window, where a number of instances in the window are replaced by the same number of incorrectly classified instances outside the window at each stage, and the growing window, where the window grows by the addition of (previously incorrectly classified) instances at each stage. The role of iteration in ID3 will be discussed in Section 4.5.

Quinlan (1986) describes the initial window as being chosen at random and mentions only the 'growing window' method of selecting a replacement window. (It would seem that the fixed window method was discarded fairly early in the development of ID3, possibly because of the looping problem mentioned in Section 4.5.)

In the original version of ID3,[FpY'Xwo classes were allowed for any induction task, but Quinlan (1986) points out that this restriction has been removed in most later systems and it is ignored in the remainder of this paper. The original reason for the restriction was no doubt the roots of ID3 in research into concept learning, where the two classes could naturally be regarded as containing positive and negative instances of a concept to be learnt.

### 2.4 Developments of ID3

The principal developments from the original ID3 algorithm are reviewed by Quinlan (1986), from which the 'family tree' shown in Figure 3 is taken.

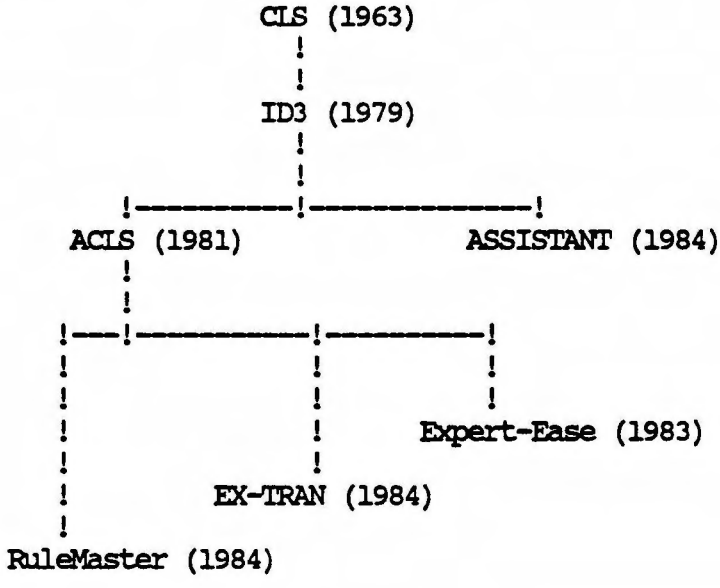



Figure 3

This figure also shows the line of descent of ID3 from Hunt's Concept Learning System (Hunt, Marin and Stone, 1966) which constructs a decision tree that attempts to minimize the cost of classifying an object.

In describing his own work, Quinlan comments: "ID3 ... is one of a series of programs developed from CLS in response to a challenging induction task posed by Donald Michie .... it embeds a tree-building method in an iterative outer shell, and abandons the cost-driven lookahead of CLS with an information-driven evaluation function".

Quinlan goes on to describe the other systems shown in the diagram as follows: "ACLS (Paterson and Niblett, 1983) is a generalisation of ID3. CLS and ID3 both require that each property used to describe objects has only values from a specified set. In addition to properties of this type, ACLS permits properties that have unrestricted integer values ....

ASSISTANT (Kononenko, Bratko and Roskar, 1984) also acknowledges ID3 as its direct ancestor. It differs from ID3 in many ways .... ASSISTANT further generalizes on the integer-valued attributes of ACLS by permitting attributes with continuous (real) values. Rather than insisting that the classes be disjoint, ASSISTANT allows them

to form a hierarchy, so that one class may be a finer division of another. ASSISTANT does not form a decision tree iteratively in the manner of ID3, but does include algorithms for choosing a 'good' training set from the objects available.

The bottom-most three systems in the figure are commercial derivatives of ACLS. While they do not significantly advance the underlying theory, they incorporate many user-friendly innovations and utilities."

As well as the systems mentioned above, several other embellishments to the basic ID3 algorithm have been developed, notably 'Structured Induction' (Shapiro, 1983).

### 3. Quinlan's KRKN Experiments

Much of the initial acclaim for ID3 rested on the success of a number of experiments conducted by Ross Quinlan in the late 1970s while working in the Machine Intelligence Research Unit at Edinburgh University.

These experiments formed part of a programme of Computer Chess research conducted at the Machine Intelligence Research Unit, much of which was based on the very detailed analysis of a subgame of chess with only four pieces - the endgame with White King and Rook against Black King and Knight (denoted by KRKN). Despite the small number of pieces involved, this endgame is a complex one even by the standards of master-level chessplayers. Even deciding whether a given KRKN position is a win for White or a draw is a non-trivial task for chess masters.

The problem studied by Quinlan was not to discriminate between wins and draws but the much simpler one (for humans) of constructing a decision tree to distinguish between positions where Black is to move which are or are not 'lost in 2-ply' for the KRKN endgame.

Black to move is lost in 2-ply can be defined by:

(a) Black is checkmated, or

(b) Black is not stalemated and whatever move he makes, White can then give checkmate or take the Knight (without giving stalemate or leaving his rook <u>en prise</u>).

Although this definition may seem complicated to the uninitiated, it is trivial for chess masters (and indeed considerably weaker chess players) to determine the correct classification in every case (using methods that are predominantly pattern- not search-based).

The following diagram shows two example positions. The classification of the first is <u>lost</u> and the classification of the second is <u>safe</u>.

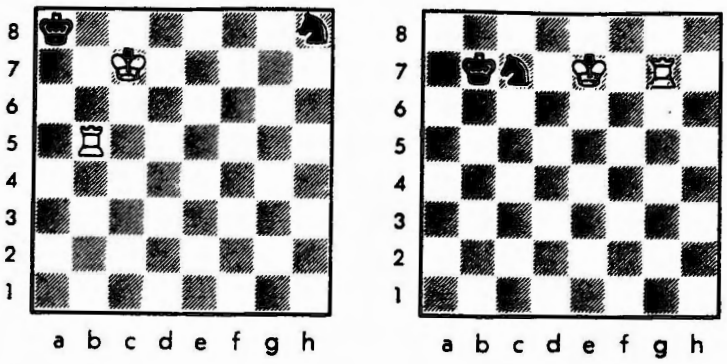



Figure 4

For this problem researchers were in the fortunate position of having a database giving (effectively) the correct classification for each of the approximately 11 million different positions with Black to move for this endgame.

The challenge was to determine whether an induction system could construct the rules determining the classification lost or safe of any legal position by analysing example positions from the database.

This problem turned out to be surprisingly difficult to solve satisfactorily. Quinlan studied seven variants of the problem, the first highly simplified, the second less so, and so on, building up to the full problem as previously stated at stage seven.

The initial step in conducting each of these experiments was to select an appropriate set of attributes. Choosing attributes such as Black King's file = 4 would have been possible, but might well have led to an excessively large number of very specific rules being generated. Instead attributes such as 'distance between Black King and White Rook = 1' were chosen (details are given in Quinlan(1979)).

Choosing an adequate set of attributes for each problem does not seem to have presented any substantial difficulties except for the seventh problem, as described below.

The large size of the training set for each problem precluded the use of the non-iterative version of the induction algorithm and the iterative version using 'windowing' was employed instead.

To choose the attribute to split on at each stage of the induction algorithm, an ad hoc complexity estimate was used, the attribute selected being the one that had the smallest value of the complexity estimate.

The complexity estimate of a given attribute was calculated by considering all possible values of the attribute in turn, calculating for each attribute value the values of S and L (the number of instances under consideration with that attribute value which have a classification of safe or lost, respectively) and summing the square root of the minimum of S and L over all values of the attribute.

A number of alternative strategies for replacing the window at each stage of the algorithm were tried, in particular:

(a) adding a limited number of wrongly classified instances, selected at random, at each iteration

(b) keeping the window size constant and replacing some instances by a random selection of wrongly classified instances at each iteration (removing those instances which had 'little influence' on forming the current set of rules).

Further details of two of the seven experiments are given below.

### 3.1 KRKN- the Third Problem

Quinlan's third problem was a variant of the original 'lost 2-ply' problem, with the considerable simplification that the board was considered to be of infinite size (from which it follows that stalemate and checkmate considerations do not apply). A much simplified version of 'lost' was also used, namely that a position was classed as lost only if the Black Knight were captured as a result of a pin, linear fork or skewer (and the same pattern of pieces on an infinite board would have the same outcome).

For this problem, seven attributes were chosen as follows:

| Attribute | Possible Values |
|---|---|
| distance from BK to BN | 1,2,3 |
| distance from BK to WR | 1,2,3 |
| distance from WK to BN | 1,2,3 |
| distance from WK to WR | 1,2,3 |
| BK,BN,WR in line | t,f |
| WR bears on BK | t,f |
| WR bears on BN | t,f |

There are 648 possible combinations of values for these seven attributes. One of these combinations is illegal under the laws of chess, leaving 647.

These seven attributes also form an 'adequate' set, i.e. no two positions with the same description belong to different classes. Hence the millions of positions in the original database can be replaced by a training set of 647 instances, each a combination of attribute-value pairs and the associated class.

Two methods of replacing the window were tried for this experiment:

(a) An expanding window

Thirty instances were chosen at random to form the initial window, with at most 10 exceptions added at each iteration.

To find a correct set of rules for all 647 instances took six iterations and 211 seconds of cpu time on a KA-10 processor (the program was written in Interlisp).

The final window contained 68 instances.

(b) Fixed window size

A fixed window size of 50 instances was taken. In this case the same correct rule set was discovered as for (a), but this required seven iterations and 224 seconds.

(It is worth remarking that the result obtained from method (b) shows that only 50 out of the 647 instances in the training set (about 7%) evidently capture the 'essence' of this third problem.)

### 3.2 KRKN- the Seventh Problem

For the seventh experiment, the full 'lost in 2-ply' problem was taken.

In this case, it did not prove possible to find a satisfactory, compact set of attributes that were 'adequate', i.e. for which the classes did not overlap. This problem was solved by the introduction of a third classification 'search' (meaning that the classification into 'safe' or 'lost' is not determined by the induced set of rules, but has to be found by subsequent tree search).

With this adjustment to the problem, i.e. a classification into 'safe','lost', or 'search', an adequate set of 25 attributes was chosen. There are 29,236 valid combinations of the values of these 25 attributes which occur for the positions in the KRKN database. The training set therefore comprised 29,236 instances, of which 75 were classified as 'search'.

An initial window of 400 instances was chosen at random and a 'growing window' approach was taken, with up to 100 exceptions being added at each iteration.

For this problem, the ID3 program was written in Pascal to increase its speed, and was run on a KL-10 processor.

Twenty iterations were needed (taking 394 seconds processor time) to reach a final decision tree.

The decision tree had 393 branches of which 11 were labelled 'search'. The final window comprised 2160 instances (approximately 7% of the training set).

Further details of the rule-induction experiments can be found in Quinlan (1979).

### 3.3 Commentary on the KRKN Experiments

The performance of the ID3 algorithm on the KRKN problems is clearly very promising. Nevertheless, it must be borne in mind that the problems solved were ones which are trivial for strong human chess-players. The difficulty Quinlan experienced in finding an adequate set of attributes for the seventh problem is surprising, especially following six successfully conducted experiments.

The KRKN problems are unusual ones in that all possible instances are available in the training set, all attribute values are known and there is no possibility of 'noise' in the data. Most real-world

problems differ from this. In particular, a complete training set is only rarely available.

Although it has been widely assumed that the rules induced in the KRKN experiments are very satisfactory (relatively simple, understandable to experts, etc.), a recent analysis by Cendrowska (1986) casts doubt on this.

The decision tree produced by ID3 (using the 'information theoretic' formula described in Section 2.3 for choosing the attributes to split on rather than the ad hoc complexity estimate referred to above) has 52 branches, each corresponding to a rule, with a total of 337 component terms. However, Cendrowska demonstrates that this can be reduced to an equivalent set of only 15 rules, with 48 component terms, which are patently at least as meaningful to subject experts as the ID3-induced rules.

In this case, it would seem that the choice of a decision tree as a representation for rules in ID3 actually obscures some of the simple relationships which are present in the data. This point will be discussed further in Section 4.6.

## 4. The ID3 Rule Induction Algorithm- A Critical Review

In the following sections the limitations of the ID3 approach are considered and some of the problems that can arise are identified.

### 4.1 Generality

The ID3 algorithm is only applicable in situations where:

(1) there is a universe of objects, each belonging to one of a set of mutually exclusive classes

(2) there is a known set of attributes which are 'adequate' to classify all the objects (i.e. no two objects with the same values of all the attributes are in different classes)

Although reasonably general, these conditions are certainly not universally applicable. In particular, ID3 cannot be used (without modification) when:

(1) the value of an attribute is unknown or uncertain (e.g. it is known to be either A or B) for some object in the training set

(2) there is 'noise' in the values of one or more attributes or in the classification of an object

(3) an adequate set of attributes cannot be found

Some restrictions in the original version of ID3 have been eased in subsequent versions, e.g. by allowing attributes to have continuous values (in ASSISTANT) or by allowing rules to have components involving inequality relations for integer-valued variables (in the Expert-Ease package). However, more fundamental restrictions remain. In particular:

(a) it is not possible to construct rules relating different properties (attributes) of an object (such as Length = Breadth or Force = Mass X Acceleration) or rules relating properties of different objects in a domain (such as patients, cultures and organisms in MYCIN), or inexact rules (such as those in MYCIN, Prospector and most of the other major expert systems)

(b) although any adequate set of attributes is formally sufficient, the induced rules are only likely to be meaningful if the attributes are intelligently chosen

(c) the crucial requirement that rules not only classify the instances in the training set but are able to predict the class value for instances not included introduces further complications.

These and other problems of the ID3 approach are discussed further below.

**4.2 Complete Versus Partial Training Sets**

There are two significantly different reasons for using an induction algorithm to induce rules from the instances in a training set: data compression and prediction. It is necessary to distinguish between two cases:

(1) when the training set is complete, i.e. includes every possible instance (with all possible valid combinations of values of the chosen attributes)

(2) when the training set is partial, i.e. is a sample from a larger set of instances, which is not available in full.

In the former case, provided the method produces a set of rules, there is a clear sense in which those rules are correct, i.e. they will produce the correct classification for any possible instance, although there may not necessarily be universal agreement that the induced rules are 'reasonable', 'natural for people', etc.

Although no more information is available after the induction algorithm has been performed, since it was all already available in the training set, the induced rules once found can then replace the training set in subsequent use, frequently giving a considerable 'compression' relative to the raw data in the training set.

The training sets used in Quinlan's KRKN experiments were complete ones (ignoring the exclusion of some instances which could never legally occur under the laws of chess), and in the seventh experiment a compression of data from 29,236 instances to 393 rules was achieved.

For case (2) it is not generally of prime importance to find rules to compress the data. Rather, the task is to find the rules which generated the data, assuming such rules exist in the form of conjunctions of 'attribute = value' components. The aim is to be able to predict the correct classification for instances not in the training set. It is this use of the induced rules which justifies the term 'induction' being applied to the process in question.

As has been pointed out previously, the induction algorithm is guaranteed to terminate (at least in the non-iterative form) with any choice of attribute to split on, provided that some adequate set of attributes can be found. In general, there are many possible rule sets that can be induced for any given training set, a trivial one being simply one rule per instance.

A different criterion then comes into consideration: not whether a set of rules can be induced, but how to ensure that the induced rules capture the underlying causality of the domain and hence have a high level of predictive power.

The distinction between cases (1) and (2) - essentially that between 'closed' and 'open' worlds - is crucial in considering induction algorithms. However, the two cases do not seem to be well differentiated in the literature, where the whole topic of prediction receives surprisingly little explicit attention.

Although Quinlan's original work was in the closed world of the KRKN endgame, the ID3 algorithm has often been applied subsequently in the open world case, which Expert System domains typically are, with only a relatively small proportion of all possible instances available.

The question of the predictive power of induced rules will be considered in Section 4.7 below.

### 4.3 Choosing an Attribute to Split On

The choice of the most appropriate attribute to split on at each stage of the basic induction algorithm is in one sense arbitrary (since any choice will still lead to a decision tree), but in practice is likely to be of crucial importance if the induced rules are to have any predictive value.

As an example, Table 2 shows four attributes recorded for each of 12 Thames Polytechnic students (eye colour, married status, sex and hair length) together with the 'class' to which each belongs- in this case whether each is a member of the Polytechnic's Rugby club or its Netball club. (For simplicity, it will be assumed that there is a strict rule that each student must belong to one and only one of these two clubs.)

| eyecolour | married | sex | hairlength | class |
|---|---|---|---|---|
| brown | yes | male | long | rugby |
| blue | yes | male | short | rugby |
| brown | yes | male | long | rugby |
| brown | no | female | long | netball |
| brown | no | female | long | netball |
| blue | no | male | long | rugby |
| brown | no | female | long | netball |
| brown | no | male | short | rugby |
| brown | yes | female | short | netball |
| brown | no | female | long | netball |
| blue | no | male | long | rugby |
| blue | no | male | short | rugby |

Table 2

One possible decision tree which can be induced from this data is Figure 5 below.

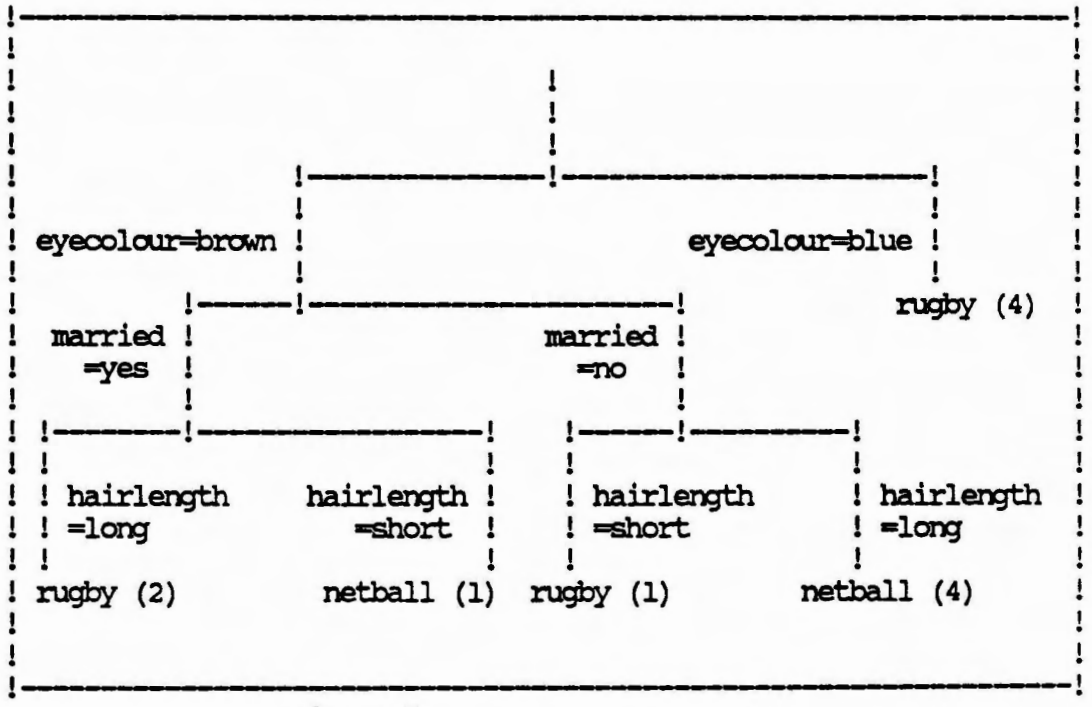



Figure 5

Although Figure 5 seems quite a reasonable way of classifying the instances in the training set in Table 2, a different choice of the first attribute to split on would have made it a great deal more compact. The simplest decision tree derivable from the training set is merely:

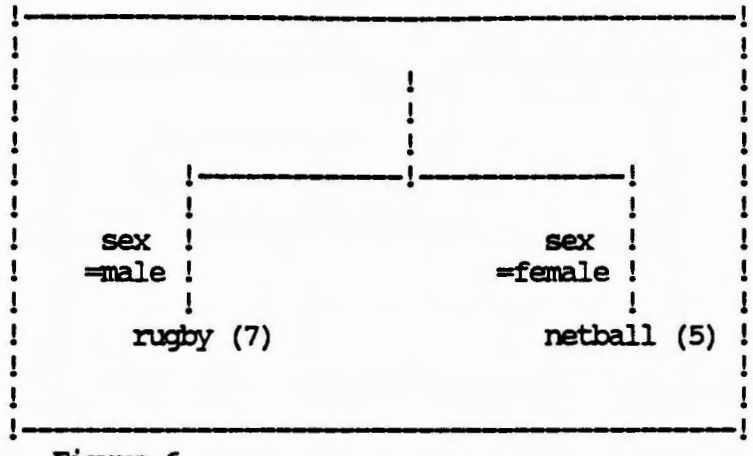



Figure 6

It looks suspiciously as if the real distinction between members of the two clubs is their sex! If this is indeed the case, Figure 6 is certain to predict perfectly for every instance and the predictive power of all other possible decision trees is likely to be extremely low (unless they include a test of the value of attribute 'sex' in every branch).

On the other hand, if there are (say) male students who are members of the netball club but who are not included in the training set, the decision tree shown in Figure 6 will certainly fail to make the correct prediction for them but the decision tree shown in Figure 5 might.

The simplicity of the decision tree shown in Figure 6 may look convincing, but whether or not it correctly captures the underlying causality of the domain from which the training set was drawn depends on the nature of the training set, in particular whether all 'critical cases' are included.

Unfortunately, in many real-life situations, where data is collected on a partially 'self-selecting' basis (e.g. from patients admitted to a particular hospital or from passers-by being interviewed in the street) it is extremely difficult to ensure that all critical cases are included as well as to avoid an unintended bias. Example cases selected by experts are clearly much more likely to include critical cases (which may occur very rarely in practice) but are extremely unlikely to avoid (statistical) bias without a great deal of care. It is clearly not likely that a training set will both be a random sample and include all necessary critical cases.

One serious fundamental weakness of the method used to select an attribute in ID3 is that it is formula-based and thus basically an 'unintelligent' process. There is nothing explicitly present in the method that prevents an attribute of little or no relevance being selected to split on in preference to one of high relevance.

Thus in a medical training set, it might be well understood by experts that the sex of the patient was likely to be of great importance, whereas (say) the patient's country of origin was of much less importance. There is no explicit way this information can be represented or used in ID3, although it could possibly be dealt with by associating costs with attributes as in CLS.

In the normal 'open world' (partial training set) case, the effect of even one bad choice of attribute to split on when constructing a decision tree can be quite substantial. As an extreme example, if a set of attributes contains one that is completely irrelevant to the classification and by chance this one is chosen to split on first, the eventual decision tree at worst may be completely meaningless and at best might well include a number of identical subtrees - one for each of the values of the irrelevant attribute.

Quinlan's use of an Information Theoretic measure to determine which attribute to split on (described in Quinlan (1986)) was specifically aimed at selecting the most relevant attribute on each occasion and seems to have proved successful in practice, but this approach could not invariably be reliable. The formula makes use of probabilities and in the open world case these are generally not known, but can only be estimated from the corresponding frequencies of occurrence in the training set.

Since as pointed out above the training set will generally not be a random sample from the (usually unknown or unavailable) set of all possible instances, estimating probabilities from frequencies will not necessarily be reliable. In any case, it is not clear why the Information Theoretic measure should give a decision tree which is either meaningful to subject experts or of particular predictive value in the 'open world' case, however valuable it may be in the 'closed world' case of a complete training set.

Quinlan (1986) quotes an example reported by Ivan Bratko's research group of a medical induction problem where subject experts had disagreed with the choice of attribute to split on ('age of patient') produced by the information theoretic measure described in Section 2.3. A similar situation is discussed by Kononenko, Bratko and Roskar (1984). The occurrence of such disagreements hardly seems surprising when a uniform formula is applied in situations where only a partial (and generally non-randomly selected) training set is available to the program.

The value of using a formula based on maximizing the expected gain of information is also not entirely clear. It has been pointed out that if the attribute 'name' or 'reference number' is included in a training set (say in a medical classification application) it is certain to be chosen to split on and, indeed, completely determines the classification. Unfortunately, it has no predictive power whatever for instances not in the training set.

Rather than looking for an alternative formula which will overcome problems such as those mentioned above, a better approach might be to avoid the use of any uniform formula for selection and instead to look for ways of including in the selection process the knowledge of subject experts concerning which attributes are in general the most important in determining the classification. Quinlan (1979) mentioned this approach, commenting 'This promising avenue has not yet been explored', but preferred to adopt an approach based on the aim of generating as simple a decision tree as possible.

The knowledge-based approach will be discussed further in Section 5 below.

### 4.4 Choice of Attributes

Choosing the attributes by means of which a set of objects in a domain are described generally involves making a selection (and thus losing some information compared with the full complexity of the real world).

Formally, the only restriction placed by the rule induction algorithm on the choice of attributes is that the set has to be adequate, i.e. that no two instances share the same values of all the attributes.

In principle, there are many different ways in which an adequate set of attributes may be chosen which will lead to an eventual decision tree (for the non-iterative version of the induction algorithm any adequate set of attributes will suffice). However, with a poor choice of attributes, a large number of rules (almost equivalent to a case-by-case tabulation) are likely to be produced with little or no predictive power (see Section 4.7 below).

Although a good choice of attributes is likely to be of crucial practical importance (at least in the 'open world' case), this may be extremely difficult to make, even for experts. Although experts are obviously unlikely to make major errors in their choice of attributes, it is perfectly possible that at least one important (but not entirely obvious) attribute may be inadvertently omitted from the set chosen or unimportant attributes may be included. Such relatively minor errors of judgment by a subject expert could easily go entirely unnoticed, with a decision tree produced that looks reasonable but in fact has little or no predictive value.

To take an extreme example, supposing that in the 'team membership' training set shown in Table 2 (Section 4.3), the crucial attribute 'sex' were omitted from the attribute set. The remaining attributes still form an adequate set, and the induction algorithm will still produce a decision tree, such as Figure 5.

However, (assuming that a student's sex is what really determines his or her team membership) it is obvious that the rules induced will have little or no predictive value.

The three rules derived from Figure 5 which conclude that the class is 'rugby' are:

```
IF   (eyecolour=brown AND married=yes AND hairlength=long) THEN
     rugby
IF   (eyecolour=brown AND married=no  AND hairlength=short) THEN
     rugby
IF   (eyecolour=blue) THEN rugby
```

whereas the single such rule in Figure 6 is:

```
IF sex = male THEN rugby
```

There is clearly no sense in which the combination of attribute values

```
(eyecolour = brown AND married = yes AND hairlength = long)
```

OR (eyecolour = brown AND married = no AND hairlength = short)
OR (eyecolour = blue)

is genuinely equivalent to sex = male, but that happens to be the case for the training set under consideration. Effectively, in this example where the attribute 'sex' is assumed to have been omitted from the training set given in Table 2, the induction algorithm has found a 'circumlocution' for a value of a crucial but omitted attribute. However, this can hardly be of general applicability or useful for predicting the class membership of instances not in the training set.

Although the problem of an important attribute being omitted is unlikely to arise with the most important attributes in a domain, it is quite probable that it will do with less important attributes and may reduce the value of the rules generated quite considerably.

The problem of finding an adequate set of attributes can also not be lightly dismissed. This might be thought to be straightforward, but Quinlan's inability to do so for his seventh problem (leading to the ad hoc introduction of a third class - 'search' - for awkward cases) is not encouraging, not only in view of the considerable experience of selecting attributes he must inevitably have gained from the previous six experiments, but also in view of the small number of chess positions (only 75 out of 29,236 instances in the training set) for which this third class had to be introduced.

**4.5 The Role of Iteration**

There is some uncertainty in the literature whether the use of iteration based on the progressive modification of a window is an essential and desirable feature of the ID3 algorithm or simply a way of coping with a large training set. Quinlan (1979) strongly suggests the latter. He remarks "We assume that the training set available to the experimenter is substantial - it may even contain all possible instances. Such a training set will usually be too large to present to the induction algorithm, requiring either too much memory or excessive computation. The approach taken ... was to select a subset of it called a window and to modify this subset in the light of the rule[s] discovered using it".

Unfortunately, there is no guarantee at all that the iteration will converge. In the case of a fixed window, since instances which have once been discarded from the window can subsequently become 'exceptions' (i.e. wrongly classified instances) and be reintroduced, it is perfectly possible that the algorithm will loop. For a growing window, O'Keefe (1983) has shown that the iteration is only certain to converge in the 'degenerate' case where the window can grow to include the full training set.

Quinlan (1986) does not mention the fixed window at all, but quotes O'Keefe's result for a growing window. He makes the interesting comment that 'this potential limitation has not yet arisen in practice'. The fact that a potential problem has not shown up in practice may indicate that human skill (in selecting appropriate attributes, choosing a suitable window size, etc.) has played a more important role in practical uses of ID3 than has been generally recognised.

Despite quite a reasonable volume of use of the ID3 algorithm in a number of research groups, no consensus seems to have arisen concerning the best way of handling the windowing. The use of iteration has frequently been dispensed with altogether, the induction algorithm being applied directly to the entire training set.

Although the use of iteration involves some risk that the induction algorithm may not terminate, when termination occurs there is at least some reason to imagine that the induced rules will have a reasonably high level of predictive power, since rules satisfactory for all the instances in the training set have been generated from only a subset of it.

**4.6 Representing Rules as Decision Trees**

Although any decision tree of the kind under consideration can be converted in a straightforward way to a set of rules, one per branch, it is not possible in the general case to convert a set of rules relating attribute values to a classification into a decision tree without introducing additional rule components.

Suppose that for some training set similar to Table 1, the underlying rules which generated the instances were:

(a) IF eyecolour = blue THEN computing

(b) IF hairlength = short THEN humanities

(c) IF sex = female THEN computing

To represent these rules in a decision tree, it would first be necessary to change two of them by introducing attributes in common with the third rule, e.g. the rules (b) and (c) might be rewritten as:

(b1) IF eyecolour = brown AND hairlength = short THEN humanities

(c1) IF eyecolour = brown AND hairlength = long AND sex = female
THEN computing

A decision tree could then be constructed, with splitting taking place on attributes eyecolour, hairlength and sex, in that order.

In this case, being forced to write the rules in the form of a decision tree leads to an adjustment of the rules into a more complex form, including components which would not otherwise be necessary. The decision tree format actively militates against the construction of simple rules with only one component term.

As a more significant example (already mentioned in Section 3.3), Cendrowska (1986) has shown that the rules for Quinlan's third KRKN problem can be rewritten in a substantially more compact form (which is still meaningful to human chessplayers), but one which does not fit naturally into the form of a decision tree. In fact, it would seem that the need to fit the rules into a decision tree format may have added unnecessarily to their complexity in Quinlan's KRKN experiments.

In such cases, it may be possible to obtain the more simple form of the rules from the decision tree, by converting the tree into rule form, one rule per branch, and then simplifying the resulting rule set. However, the need for such simplifications would itself suggest that a decision tree is not always the most effective representation to use for induced rules.

There are two further problems involved in constructing an induced decision tree:

(1) It may not always be wise to perform a full split on each attribute. Suppose, for example, in a training set drawn from a legal application one of the attributes included is 'country of birth'. There may be a legal rule specific to those born (say) in the Falkland Islands and it would be sensible to deal with all the corresponding instances early in the rule induction process. However, it would probably be most unwise to perform a full split on 'country of birth', since this would involve building (identical or almost identical) subtrees for those born in numerous other countries (Australia, New Zealand, Japan et al).

(2) Representing rules in the form of decision trees makes it impossible to obtain certain combinations of rules (except possibly by post-processing of the decision tree).

Returning to the training set shown in Table 1 (Section 2.1), a simple rule extracted from Figure 1 is

IF eyecolour = blue THEN computing

and a rule which can be extracted from Figure 2 is

IF sex = female AND hairlength = long THEN computing

These two rules were found by examining two different induced decision trees for the same training set, but there is no means by which both rules could be induced (in the form shown) in a single execution of the induction algorithm, whether a standard full split strategy or a partial split strategy is adopted.

From these examples, it would seem that developing a decision tree may not invariably be the correct course to follow and consideration should be given to the construction of algorithms which generate rules directly in a suitably compact form. Such an approach has been followed by Cendrowska (1987).

### 4.7 The Predictive Power of the ID3 Algorithm

An important distinction was drawn in Section 4.2 between two fundamentally different cases.

(a) The training set is complete and the task is to find an efficient way of discriminating amongst the various classes. For this case it is not particularly important (although it may be desirable) for the rules to be meaningful to an outside human

observer. It is enough for them to be correct and a more compact representation of the data than the original training set.

(b) The training set is a partial one which forms a sample from a much larger set of possible instances. Here the requirement is to construct a set of rules in the form of a decision tree which can then be used to predict with a high degree of reliability the class to which an object not included in the training set would belong, solely on the basis of the instances which were used to construct the decision tree.

In practical Expert System projects, case (b) is likely to be by far the more common situation.

For such an important topic there is remarkably little explicit discussion in the ID3 literature of the predictive power of the induced decision trees. This may be because Quinlan's KRKN experiments made use of a complete training set where the question of prediction did not arise in any significant way.

Paradoxically, one of the weaknesses of ID3 is that the method is too powerful: it can generate rules where they do not exist. For example, it could be used by the inexperienced to find rules for weather prediction from tables of temperature, rainfall etc. for successive days. For small training sets, it may well be possible to find an adequate set of attributes but the induced weather-forecasting rules will be meaningless.

Returning to the training sets given as Tables 1 and 2 in previous sections, it is surely obvious that whether a student takes a computing course or belongs to the netball team cannot really be related to the colour of his or her eyes. Finding rules which make such evidently spurious connections is of no predictive value whatsoever - and yet there is nothing in the ID3 algorithm to stop spurious attributes being recorded and meaningless relationships discovered.

Naturally, the skill (and commonsense) of the users of ID3 and related systems can be relied on to prevent such obviously worthless sets of induced rules being either constructed or reported in the literature. However, there are likely to be more subtle cases where spurious rules are generated which may be far less easy to detect, especially if the induced decision tree is large.

For a given training set, there are many possible decision trees which can be constructed. However, most of these are likely to have a low predictive value. To take an extreme case, one decision tree which can be constructed from a 1000 instance training set with 8 attributes per instance will have 1000 branches, each with eight components- one instance per rule! This tree will obviously have no predictive value at all (unless the training set is complete). Other selections of decision tree from the large number of possibilities may vary considerably in their predictive power.

There is a clear analogy with problems such as those in so-called 'intelligence tests', a typical example of which is:

find the next term in the sequence 1,4,9,16, ....

Although presented as find the next term, any number is a valid answer and there are infinitely many formulae which will generate the sequence (obtained, for example, by fitting polynomials of progressively higher degree).

One possible formula for the nth term is:

$$(-5n^4+50n^3-151n^2+250n-120)/24$$

which gives a value of 20 for the fifth term - an answer which is virtually certain to be marked as simply 'wrong' by the examiner of the intelligence test!

Although the uninitiated may regard the task as that of finding a (or the) solution, this is in fact trivial and the real task involved is to find the right solution, in the above case the formula which was in the mind of the problem setter.

In the case of automatic rule induction, the task is to find the set of rules which generated the data, i.e. which capture the underlying causality of the domain.

To do this would unavoidably seem to involve knowledge of what constitutes a reasonable rule in the domain under consideration (just as in a Physics experiment, a linear or quadratic relationship may be acceptable as a reasonable outcome by a researcher, but a twelfth-order polynomial relationship would not be). However, there is no facility in ID3 for specifying what constitutes a 'reasonable' rule.

In general, the induced decision tree is only likely to have high predictive power if:

(i) the attributes are well-chosen

(ii) a good choice is made of the attribute to split on at each stage

(iii) the training set is large and includes critical cases.

Some would add to this list the requirement that the rules should be meaningful to a human expert, however this does not seem necessarily to follow (even though meaningfulness is obviously desirable for other reasons).

From the discussion earlier in this section, it is apparent that each of conditions (i) to (iii) is problematic.

A further feature of the ID3 algorithm which may lead to rules of questionable predictive power is the necessity of constructing a decision tree which 'covers' all the instances in the training set. Whereas some branches may each correspond to (say) 20 per cent of the instances, and represent meaningful rules, other branches may correspond to only 1 or 2 instances each and be of little or no predictive value. A better approach may be to seek to identify a number of 'special cases' in the training set which are ignored when the decision tree is constructed.

### 4.8 Decision Trees and Expert Systems

Although rule induction is often described as a means of automatically generating the rules for an Expert System, it is not clear that the relationship between the rules induced by ID3 and those in 'traditional' Expert Systems such as Dendral and MYCIN is particularly close, for two reasons.

(1) Rules in Expert Systems are frequently inexact, and this is typically represented by a numerical certainty factor. Rules induced by ID3 are invariably exact.

(2) Only certain kinds of rule can be induced by ID3. In general the corresponding 'rule language' is not expressive enough to represent many of the rules typically found in Expert Systems (even ignoring the problem of certainty factors), for example those rules in MYCIN which relate properties of different contexts.

It could also be argued that one valuable use of rule induction should be to enable 'patterns' to be found in data and that this also points to the need to be able to find inexact rules. However it is not clear how the ID3 algorithm could be modified to allow this.

### 4.9 Sensitivity

There is little information available on the sensitivity of ID3 and its derivatives. It is at least possible that in the iterative versions of the algorithm such factors as the window size and the criteria by which instances are added to/deleted from the window may substantially affect the final induced decision tree. Indeed, as it is known that looping may occur, it may well be that certain choices of window size, replacement criteria etc. lead to infinite looping whilst other not very dissimilar choices lead to a successfully generated decision tree. The question of sensitivity merits more detailed analysis before the ID3 algorithm is accepted into common use.

### 4.10 Dealing with Noise

The basic ID3 algorithm assumes that all attribute values are known and exact and that all the classifications are correct. In practice, it is usual for there to be 'noise' in the data, i.e. for attribute values to be unknown or incorrect, or for classifications to be in error.

The effect of noise on the ID3 algorithm has been analysed by Quinlan (1986), who has developed various techniques for dealing with it.

An alternative approach might be to take a lesson from the problem of fitting curves to points on a (two-dimensional) graph, for which there are well-established exact techniques.

However, in fitting a curve to data points obtained from a scientific experiment (where noise in the form of measurement and other experimental errors is almost always present), these techniques are not used. It is assumed that the presence of noise is

the normal case and an entirely different technique is used, frequently that of fitting a 'best straight line' through the data by the method of least squares.

Two important ingredients of this approach are:

(1) a loose fitting of the data

a loose fit is considerably more 'robust' than an exact one and of far greater predictive value in practice (in the unlikely event that the data is free of noise the predictive power would be less than that of an exact fit but this possibility can normally safely be discounted)

(2) the ignoring of outliers

i.e. ignoring a small number of data points which are so far from the general 'trend' of the others that including them would substantially contaminate the result. Including outliers in the calculations can achieve little and may simply be 'fitting the noise'in the data.

Returning to ID3, an alternative to Quinlan's methods of dealing with noise which may merit further consideration would be to look for techniques of loose fitting of rules to a given training set and for ways of identifying outliers as special cases to ignore when constructing a decision tree. As an example of the second of these, the 75 instances which had to be classified as 'search' in Quinlan's seventh KRKN experiment might have been better regarded as special cases and ignored when building the decision tree.

**4.11 Types of Attribute**

It is helpful to distinguish between (at least) two different types of attribute, those which are (a) nominal and (b) ordinal. Most of the attributes included in a training set are likely to be of one or other of these types, and both can pose problems either in the usual equality relations of ID3 or in the inequalities permitted in some of its extensions.

(a) Nominal Attributes

These are attributes which take a (usually small) unordered finite set of possible values, such as sex (with values male and female) or programming language used (with values Fortran, Algol, Cobol, Prolog, Lisp, ....). Although the values of such attributes may sometimes be represented by integers for convenience (e.g. country-of-birth = 1,2,3,4 or 5),it is obviously meaningless to use these numbers in conjunction with inequality relations (such as country-of-birth>3). A purely formal exception arises when some condition holds whether say country-of-birth = 3, 4 or 5, in which case the inequality country-of-birth>3 can be used as a means of combining rules.

An important special case of a nominal attribute is one such as name or reference number where every instance has a different value of the attribute. In a complete training set, such an attribute can be used as a unique key (e.g. IF reference-number = 287 THEN

class = A). In a partial training set, rules including this kind of attribute will have no predictive power whatsoever and they should never be chosen to split on.

(b) Ordinal Attributes

Ordinal attributes are those such as age (in years), where the possible values form an ordered set. Performing a full split on such attributes, and thus generating rules with components such as age = 23, does not seem desirable since no single value of age is likely to be particularly meaningful. An inequality relation such as age>70 may be meaningful, but a superior alternative may be to avoid using such attributes altogether and instead to analyse the values occurring in the training set, leading to a grouping into (say) child, young, middle-aged and old, which effectively converts an ordinal into a nominal attribute.

(c) Other Kinds of Attribute

Nominal and ordinal are by no means the only possible categories of attribute although they are likely to be the most common in practice. An example of an attribute which is somewhere between nominal and ordinal occurs in Quinlan's KRKN experiments, where all distances of Kings from other pieces which were greater than two were replaced by the value '3' (thus making 3 the label for all values from 3 to 7 and circumventing the problem of the absence of inequality relations in the original version of ID3).

**4.12 Summary**

From the above discussion it is evident that there are still considerable areas of difficulty associated with the use of the ID3 algorithm. However, these do not seem to have shown up in the use of the system in a number of practical applications.

The explanation for this apparent conflict is likely to be that the skill and experience of the fairly small number of groups which have so far used ID3 and its derivatives has enabled them to bypass potential areas of difficulty.

This skill could be applied in several ways, e.g. by selecting suitable problems to tackle, by making a good choice of attributes or by choosing an appropriate window size.

Paradoxically, as ID3 becomes better established, it is likely to be used increasingly by those without the experience or technical knowledge to avoid the pitfalls of the method. In unskilled hands, ID3 can easily produce rules that are entirely meaningless and of no predictive value whatsoever.

This situation is analogous to the position in the field of Statistics. The use of statistical techniques by skilled practitioners is of undoubted value, but their use by those without suitable experience and theoretical knowledge can give results which are seriously misleading.

The availability of increasingly sophisticated computer-based statistics packages to progressively less skilled users is already a

matter of justified concern. This may be an appropriate point to issue such a warning about the passing of rule induction algorithms into unskilled hands.

## 5. An Alternative Approach to Rule Induction: the 'Rule Induction Assistant'

ID3 makes use of expert knowledge only in the initial selection of classifying attributes and (possibly) in the choice of a training set of instances. The rule building process is then entirely automatic.

This fully automatic construction of rules ignores the possibility that the domain expert has valuable knowledge which could be brought to bear on the construction of rules: for example, he or she may already know some (although not all) of the rules, may know that some attributes are more important or 'interesting' than others or that some combinations of attribute values are meaningless, likely to be significant etc.

An alternative approach which is currently being investigated by the author aims at developing a "rule induction assistant" system, i.e. a semi-automated system which captures such additional expert knowledge explicitly and makes use of it to guide the construction of decision rules with high predictive power.

The envisaged system places the domain expert in an interactive dialogue with the system during which he or she is able to specify a number of constraints on the process of generating the rule set. Some of these constraints are specified at the beginning of the interactive session, others can be added interactively as the session progresses (and the expert's own knowledge of the domain possibly increases).

The system enables the expert to experiment by generating a number of possible rule sets (some but not all in the form of decision trees, for which the user may also specify the criteria for selecting which attribute to split on at each stage). It is assumed in this that the expert, although not having knowledge of all the underlying rules in detail, may still have sufficient insight to accept or reject a rule hypothesised by the system.

It is envisaged that a 'rule induction assistant' system would provide the expert user with a range of facilities including the following:

(a) <u>Concerning Rules</u>

(i) to specify a rule (either known in advance or found during system execution, e.g. as a branch of some decision tree) for permanent retention

(ii) to specify that certain combinations of attribute values should never occur together in a rule

(iii) to specify a 'reasonable', 'unreasonable' or 'preferred' form for a rule (e.g. rules with only one 'attribute =

value' component are preferred, or rules which include certain combinations of attributes are likely to be significant/meaningless, etc.).

(b) Concerning Attributes

(i) to specify dependencies between attributes (e.g. moustache = yes => sex = male)

(ii) to specify which attributes are nominal, ordinal, etc.

(iii) to delete an attribute from each instance in a training set

(iv) to add a new attribute to all instances in a training set (with a value derived from the value of one or more existing attributes)

(c) Concerning Instances

(i) to delete an instance from the training set (e.g. if it is considered to be too 'noisy')

(ii) to specify that an instance is a 'special case', i.e. should be ignored when generating a rule set

(iii) to correct an attribute value or a classification in an instance (if an error is found)

(iv) to provide a missing attribute value

(d) Concerning the Choice of Attribute to Split On

To specify that an attribute should be given a high or a low priority for selection (to override the use of a selection formula), should never be selected (e.g. 'patient's name'), or should be selected only when there are no alternatives (e.g. 'number of children').

(Some of these facilities, e.g. (c)(i) - (c)(iv) are of course already available to the user of ID3, but not interactively.)

The information provided by the system to the expert would need to include information on the number of instances 'covered' by each branch of a decision tree and the facility to reject a rule which covered too few instances (in the expert's judgement), even if this meant that not all instances were covered by the final set of induced rules. (It is surely preferable to generate an incomplete set of rules with high predictive power for instances to which they apply, rather than a complete set of rules, some of which are of very low predictive power.)

As a practical means of identifying noise, the additional facility would be included to search for simple rules (say, with one or two components only) which would hold in a high percentage of cases, but not all.

## 6. Conclusions

The induction approach to constructing rules from sets of examples is of considerable importance as a practical means of avoiding the usual knowledge acquisition 'bottleneck' encountered when developing Expert Systems.

A rule induction algorithm which has been widely used and advocated in its various forms is the ID3 algorithm, originally developed by Ross Quinlan. In the hands of experienced researchers, such as those at the Turing Institute in Glasgow and Quinlan himself, considerable practical successes have been reported with this algorithm.

However, the analysis in previous sections indicates that the method has many potential pitfalls and certainly cannot be used 'blindly' by the inexperienced.

It is reasonable to hypothesise that in the hands of experienced researchers, both they and perhaps the subject experts they work with are able to bring additional knowledge to bear on the process of rule induction which is not explicitly present in the ID3 algorithm itself.

With the increasing acceptance of the rule induction approach, it is likely that the ID3 algorithm will come to be used by those with considerably less skill than previously and in such a situation there is an obvious risk that a number of highly unreliable rule sets will be produced and it is entirely possible in the context of a large Expert System that even serious errors will go undetected for a considerable time.

If this problem is to be avoided, it is important that both the inherent limitations of the ID3 algorithm and the additional skills that need to be brought to bear by its users be explicitly identified and the preceding critical analysis is presented in this spirit.

One possible means of improving the value of the ID3 approach would be to provide facilities for domain experts to specify additional information (about crucial attributes, etc.) to the induction system interactively, and this possibility was discussed briefly in Section 5.

Even in applications where (for whatever reason) rule induction is not directly suitable on its own, it is still important to bear in mind the possibility of a 'hybrid' approach where induction is used to provide a first approximation to a set of rules, which can then be refined by conventional knowledge acquisition and refinement techniques.

## 7. References

Buchanan, B.G., Sutherland, G.L. and Feigenbaum, E.A. (1969), Rediscovering Some Problems of Artificial Intelligence in the Context of Organic Chemistry, In Meltzer, B. and Michie, D. (eds.), Machine Intelligence 5, Edinburgh University Press, pp.253-280.

Cendrowska, J. (1986). Personal communication.

Cendrowska, J. (1987), PRISM: An Algorithm for Inducing Modular Rules, Open University, Mathematics Faculty Technical Report.

Hunt, E.B., Marin, J. and Stone, P.J. (1966), Experiments in Induction, Academic Press.

Kononenko, I., Bratko, I. and Roskar, E. (1984), Experiments in Automatic Learning of Medical Diagnostic Rules, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia.

Michalski, R.S. and Chilausky, R.L. (1980), Knowledge Acquisition by Encoding Expert Rules Versus Computer Induction From Examples: A Case Study Involving Soybean Pathology. Int. J. Man-Machine Studies, 12, pp.63-87.

O'Keefe, R.A. (1983), Concept Formation from Very Large Training Sets, In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Morgan Kaufmann.

Patterson, A. and Niblett, T. (1983), ACLS User Manual, Glasgow: Intelligent Terminals Ltd.

Quinlan, J.R.(1979), Discovering Rules by Induction from Large Collections of Examples. In D.Michie (Ed.),'Expert Systems in the Microelectronic Age', Edinburgh University Press, pp.168-201.

Quinlan, J.R. (1983), Learning Efficient Classification Procedures and their Application to Chess Endgames, In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Company.

Quinlan, J.R. (1986), Induction of Decision Trees, Machine Learning, Vol.1, No.1, pp.81-106.

Shapiro, A.D. (1983), The role of Structured Induction in Expert

Systems, Ph.D. Thesis, University of Edinburgh.

Shortliffe, E. (1976), Computer-Based Medical Consultations: MYCIN, Elsevier, New York.

# ITRULE : An Information Theoretic Rule-Induction Algorithm

Rodney M.F.Goodman and Padhraic Smyth
Department of Electrical Engineering
California Institute of Technology, 116-81
Pasadena, CA 91125

In this paper we propose a general information-theoretic model for rule-based systems and describe an algorithm which can induce an optimal set of concepts or rules from a set of examples. This paper is an extension of a companion paper [1] which analysed the information theory model in the more restricted problem domain of deriving classification decision trees from a set of examples. We show in this paper that previous work in this area has concentrated on learning single concepts from examples and is classification-oriented. Motivated by the need for an induction method which can derive a more flexible set of rules for expert systems we define a non-parametric information-theoretic methodology. This information-theoretic representation is shown to be compact, mathematically consistent and very powerful. The main advantage of our approach over previous algorithms is in the flexibility of representation, allowing the algorithm to learn many different concepts.

## The generalised rule induction problem

The ability to extract general information from a set of examples is a fundamental characteristic of knowledge acquisition. The term *example* is taken to mean a feature vector of attribute values where the values are assumed to be either categorical or discrete numerical. Usually in this notation each feature vector has another component called a classification which contains the class label for that example. However for our purposes we treat the classification as just another attribute, albeit an important one. The *set* of specific examples can be thought of as a large table of these feature vectors, where 'large' means statistically sufficient for whatever computations may be required. The *general information* we wish to extract from these examples is taken to be in the form of production rules, i.e.

if *condition A* then *condition B* with probability *p*

where we allow the antecedent $A$ to be the conjuction of several simple conditions and the consequent $B$ is restricted to being a single simple condition. A simple condition is defined as 'attribute X takes on value y'. The rationale for choosing this representation is based on both theoretical and practical considerations. Theoretically it has been postulated that such rule representations are a good model for knowledge representation [2] while from a practical viewpoint such rules can be used as the knowledge-base for an expert system.

In this paper we consider the problem of obtaining the best set of rules given a set of specific examples. What is new is the fact that we do not restrict the consequent clause in our rules to be the classification attribute, i.e. we are interested in attribute-attribute rules as well as attribute-class rules. Previous work in rule-induction has concentrated on classification-type rules where the consequent clause only involves the class variable. However in data-driven applications a more flexible and general approach is desirable. One wishes to have rules relating attributes. In this sense, by treating the class as 'just another attribute', the *generalised* rule-induction problem is formulated. The more standard classification approach is seen to be a special case of the general problem.

Consider an example from the animal domain (mammals actually). Say we have a large list of animals (our example set), each animal being described in terms of various attributes such as size (large or small), colour, number of legs, dangerous (yes or no), species, etc. A classification-type of induction algorithm might induce rules such as

If *the animal has no legs* then *it is a snake* with probability p

A more general induction algorithm would induce rules such as

If *the animal is large* then *it has 4 legs* with probability p

in addition to the classification rules. The advantages of the more general approach are obvious in terms of providing a more accurate representation of the important concepts about the domain from which the examples are taken.

For the purposes of this paper we limit our attention to selective induction (where the induction algorithm retains the same representation as given by the examples) as opposed to constructive induction.

## Earlier work in the area of rule induction

A good taxonomy of automatic induction algorithms is given in Cohen and Feigenbaum [3]. These algorithms can loosely be categorised into two main areas ; those which use symbolic manipulation techniques and those which use statistically-oriented techniques. A good example of the former would be the AQ11 algorithm of Michalski and Larson [4] which achieved success in the domain of plant disease diagnosis [5]. Typically these algorithms examine the examples *sequentially* and refine what is known as the rule space until a set of general rules covering the examples are arrived at. However noisy examples (as in the case where not all but nearly all large animals in the domain have 4 legs) are not easily handled by the symbolic approach. In addition the algorithms are computationally unattractive. Consequently their use has been limited to research-oriented endeavours rather than practical applications such as knowledge acquisition for expert systems.

Methods which can termed as statistical, exploit *average* properties of the example set. Perhaps the best known of such techniques is Quinlan's ID3 algorithm [6] based on information theory. The ID3 algorithm derives an efficient classification tree for the example

set. This algorithm has been used in several practical applications [7,8]. Recent work has shown that the basic strategy of the algorithm is theoretically well-founded [9]. An interesting experiment which we carried out was to use the information measure to derive hierarchical edge detectors [1]. Because we formulated the problem in a classification framework, the algorithm in essence was able to learn basic edge detection capabilities. In addition the experiment yielded new domain-specific information as to which attribute was the single most important edge discriminator.

### The information-theoretic approach

In this paper we intend to view the generalised induction problem from a statistical viewpoint, using information theory techniques in particular. To begin with we are given a sample set of data, in the domain or universe of interest (e.g. mammals), i.e. a list of examples from this domain described in terms of attributes. We assume that the size of the sample set is sufficiently large to allow us to estimate statistical parameters to some required degree of accuracy - we need not be overly concerned with the sampling theory aspects of the problem.

The attributes ('class' included) can be viewed as random variables and, as assumed earlier, are discrete-valued. The generalised rule induction problem can then be reformulated from this random variable viewpoint - in some sense we want to 'capture' all the important interdependencies among the set of random variables. So the best set of rules should (somehow) be the most important dependency relations between the random variables - but how can we quantify this ? If $p$ is near, or equal to, 1, does that mean that the rule is good ? Not if one considers the example

If *the animal is a kangaroo* then *it does not drink beer* with probability = 1

Although $p = 1$ this cannot be viewed as a good rule, at least in most parts of the world. It is reliable but it is simply not useful. On the other hand a rule such as

If *the animal is red* then *it is a bird* with probability = 0.6

is quite unreliable but in fact it could be quite useful. Hence whatever measure we choose to represent the 'goodness' of a rule must at least be a function of the usefulness and the reliability of the rule, although we have not specified how to measure either parameter.

Consider the average mutual information measure between 2 random variables A and B, I(A;B), as defined by Shannon [10]. This measure has certain interesting properties. The definition can be given as

$$I(A;B) = H(A) - H(A|B)$$

i.e. the average information that A provides about B is our original average uncertainty about A ($H(A)$) less our remaining average uncertainty about A given B ($H(A|B)$). Another way to put it is that it is our average *decrease* in uncertainty about A once we know

B. $H(A)$ is the entropy function, a measure of how uncertain we are about the value of A.* The information measure was originally defined by Shannon in 1948 and has found almost exclusive (but extremely profitable) use in the area of communication theory. Nonetheless this measure has more general implications than just measuring the information sent across a channel.

We propose that the information measure is a natural and intuitive measure for the 'goodness' of a rule. Further we argue that it is powerful in terms of induction properties while being consistent and mathematically rigorous from a statistical viewpoint. As we stated previously, rules can be viewed as defining the relative dependency of two random variables. The information measure $I(A;B)$ is a measure of this dependency. Independent variables have zero mutual information ($I(A;B) = 0$), while completely dependent variables have the maximum amount of information possible, i.e $I(A;B) = H(A) = H(B)$. But as we saw in the kangaroo example, dependent variables do not necessarily yield good rules. Consider the information contained in this rule. The information is the decrease in our uncertainty about the variable 'beer-drinker' (yes/no), in the animal domain. But $p_{yes} = 0$ and $p_{no} = 1$ so that our initial uncertainty is zero, i.e. we are certain that animals do not drink beer. Hence knowing whether the animal is a kangaroo or not provides us with zero information. The rule is useless according to our measure - just the result we desired. If we look at the basic definition of $I(A;B)$ above we can identify in a general sense that the $H(A)$ term can be a measure of the usefulness of a rule (as it pertains to A), while the equivocation or noise term ($H(A|B)$) is a measure of the unreliability of the rule.

One of the fundamental criteria for any worthwhile induction mechanism is that it must have the property of *generalisation* (see Michalski [13]). Consider for example the rules

If *the animal is brown and it can fly* then *it has wings*

and

If *the animal can fly* then *it has wings*

Clearly the second rule is preferable to the first since it is more general. Can one measure this preference quantitatively ? The information measure has the property of additivity, i.e.

$$I(A;B,C) = I(A;B) + I(A;C|B)$$

where $I(A;B,C)$ is the average information provided about A by both B and C together, and $I(A;C|B)$ is the average information provided about A by C, given that we know B already. This latter quantity we can call the relative information of C to A given B. If this is zero, then the more specific rule contains no information *relative* to the more general rule. Hence the information provided by the variable 'is-brown' about 'has-wings' is zero, given that we already know that 'can-fly' is true.

---

* $H(A) = \sum_{i=1}^{n} p_i . \log \frac{1}{p_i}$ where A can take on $n$ values and $p_i$ is the probability of the $i$th component.

| antlers | wings | dangerous | legs | size | dog | reindeer | snake | bird | probability |
|---|---|---|---|---|---|---|---|---|---|
| yes | no | no | yes | large | no | yes | no | no | 0.10 |
| yes | no | yes | yes | large | no | yes | no | no | 0.01 |
| yes | no | no | yes | small | no | yes | no | no | 0.03 |
| no | no | yes | yes | large | yes | no | no | no | 0.16 |
| no | no | yes | yes | small | yes | no | no | no | 0.03 |
| no | no | no | yes | large | yes | no | no | no | 0.09 |
| no | no | no | yes | small | yes | no | no | no | 0.08 |
| no | no | yes | no | large | no | no | yes | no | 0.18 |
| no | no | yes | no | small | no | no | yes | no | 0.07 |
| no | no | no | no | large | no | no | yes | no | 0.01 |
| no | no | no | no | small | no | no | yes | no | 0.02 |
| no | yes | no | yes | large | no | no | no | yes | 0.01 |
| no | yes | yes | yes | large | no | no | no | yes | 0.06 |
| no | yes | no | yes | small | no | no | no | yes | 0.15 |

**Figure 1 : list of prototypes and probabilities**

In summary, the information-theory model provides a consistent measure of the 'goodness' of a rule and we demonstrated that it is clearly an appropriate model for the induction problem. Unlike many computational methods based on statistics the model is non-parametric, requiring no restrictive statistical-model assumptions.

### The rule induction algorithm

Having demonstrated the utility of the information theory approach in a general sense we present initial results from a new algorithm called the ITRULE algorithm (Information-Theoretic Rule induction algorithm). ITRULE induces the best set of rules from from an example set where best is defined in an information-theoretic sense.

The earlier work of Quinlan [6], ourselves [1,9] and others [11] was based on algorithms which when given a set of training examples ultimately produce a decision tree for classifying the examples. The disadvantage of this approach lies in the restrictive format of the tree structure. While it is true to say (as pointed out by Bundy et al. [12]) that the tree represents a form of rule structure, this structure may be too limited for applications such as deriving rule bases for expert systems where a more flexible data-driven representation is generally required.

The ITRULE algorithm derives a set of rules rather than a decision tree. A special case of ITRULE, where it focuses on the class attribute alone (in terms of consequent clauses) would be equivalent to the decision tree algorithms.

**ITRULE.1 output : the most informative set of 20 rules**

rule # 1 If the animal does not have legs then it is a snake with probability 1.00

rule # 2 If the animal is a snake then it does not have legs with probability 1.00

rule # 3 If the animal has wings then it is a bird with probability 1.00

rule # 4 If the animal is a bird then it has wings with probability 1.00

rule # 5 If the animal has antlers then it is a reindeer with probability 1.00

rule # 6 If the animal is a reindeer then it has antlers with probability 1.00

rule # 7 If the animal is dangerous and it is not a dog then it does not have legs with probability 0.93

rule # 8 If the animal is dangerous and it is not a dog then it is a snake with probability 0.93

rule # 9 If the animal has legs then it is not a snake with probability 1.00

rule #10 If the animal is not a snake then it has legs with probability 1.00

rule #11 If the animal does not have wings then it is not a bird with probability 1.00

rule #12 If the animal is not a bird then it does not have wings with probability 1.00

rule #13 If the animal does not have legs and it is large then it is dangerous with probability 0.96

rule #14 If the animal is large and it is a snake then it is dangerous with probability 0.96

rule #15 If the animal has legs and it is not a dog then it is not dangerous with probability 0.94

rule #16 If the animal is not a dog and it is not a snake then it is not dangerous with probability 0.94

rule #17 If the animal is dangerous and it has legs then it is a dog with probability 0.90

rule #18 If the animal is dangerous and it is not a snake then it is a dog with probability 0.90

rule #19 If the animal does not have wings and it is not a dog then it does not have legs with probability 0.67

rule #20 If the animal is not a dog and it is not a bird then it does not have legs with probability 0.67

**Figure 2**

ITRULE.1 is an initial implementation of the general ITRULE algorithm. The approach taken was to simulate an example set using the prototype examples of figure 1. The domain consists of the set of animals [dogs, reindeer, snakes, birds] described in terms of the attributes *antlers, wings, dangerous, legs, size.* The number of examples generated for each prototype is proportional to its probability. These probabilities were chosen rather arbitrarily, e.g. large snakes are much more likely to be dangerous than not dangerous.

The results of the algorithm on this example set are shown in figure 2. The induced set of rules represent the twenty most informative rules, listed in order from most informative downwards. The are seen to correspond well with what rules a human might consider important, capturing all the important concepts such as 'birds have wings,' 'snakes don't have legs,' etc. The results on this simple problem are quite encouraging.

## Conclusion

Having defined the problem of generalised rule induction, we proposed Shannon's information measure as a natural and coherent model from which to build a solution. We showed that the measure has desirable properties both as a measure of the goodness of a rule and as a basic induction mechanism. Initial results with the ITRULE.1 algorithm show that the measure yields rules which are intuitive and common-sense to humans. Current and future work is to build on the information theory principle and develop much more powerful induction algorithms, addressing such problems as dealing with example sets which, in a statistical sense, are not large.

## References

[1] R.M.F.Goodman and P.Smyth, 'Learning from examples using information theory', accepted for presentation at the 2nd AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, October 1987.

[2] A.Newell and H.A.Simon, *Human Problem Solving*, Englewood Cliffs, N.J : Prentice Hall, 1972.

[3] P.R.Cohen and E.A.Feigenbaum, *The Handbook of Artificial Intelligence, Volume 3* William Kaufmann Inc., 1982.

[4] R.S.Michalski and J.B.Larson, 'Selection of most representative training examples and incremental generation of VL1 hypotheses', *Report No. 867*, Computer Science Department, University of Illinois, 1978.

[5] R.S.Michalski and R.L.Chilausky, 'Learning by being told and learning from examples', *International Journal of Policy Analysis and Information Systems* 4, 125-161, 1980.

[6] J.R.Quinlan, 'Learning efficient classification procedures and their application to chess endgames', *Machine learning : an artificial intelligence approach*, R.S.Michalski, J.G.Carbonell and T.M.Mitchell (editors), Palo Alto, CA. : Tioga, 1983.

[7] D.Michie, S.Muggleton, C.Riese and S.Zubrick, 'RuleMaster - a second generation knowledge engineering facility', *Proceedings of the First Conference on Artificial Intelligence Applications*, Denver CO, December 1984.

[8] J.D.Stuart, S.D.Pardue, L.S.Carr, D.A.Feldcamp, 'TITAN : an expert system to assist in troubleshooting the Texas Instruments 990 minicomputer system', *Radian Technical Report ST-RS-00974*, Radian Corporation, Austin TX, December 1984.

[9] R.M.F.Goodman and P.Smyth, 'An information theoretic approach to decision tree design', presented at the 1986 IEEE Int. Symp. Inform. Theory, Ann Arbor, Michigan, October 1986.

[10] C.E.Shannon, 'A Mathematical Theory of Communication', *Bell Syst. Tech. Journal*, vol. 27, (pt 1) pp. 379-423, (pt 2) pp. 623-656.

[11] L.Breiman, J.H.Friedman, R.A.Olshen and C.J.Stone, *Classification and regression trees*, Belmont, CA. : Wadsworth, 1984.

[12] A.Bundy, B.Silver and D.Plummer, 'An analytical comparison of some rule-learning programs', *Artificial Intelligence* ,27, 1985, pp.137-181.

[13] R.S.Michalski, 'Pattern recognition as rule-guided inference',*IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-2, pp 349-361, 1980.

# Student Modeling: Learning a Function from Examples while Knowing a Close Function

Oded Maler

Department of Applied Mathematics
Weizmann Institute
Rehovot 76100, Israel

**ABSTRACT**

In this paper we discuss the problem of student modeling, and give it an abstract formulation. Student modeling is the process of finding out what the student "really" thinks or knows about the subject matter. The extraction of the student's misconceptions is a crucial component of any effective educational system, regardless of whether it is based on humans or machines. We regard student modeling as similar to machine learning from examples, where some background knowledge is available. When trying to infer the student's model from examples, we may exploit our knowledge of the correct model, and of the way it had been transferred to the student. It is likely that what the student has in mind is related to what he or she had been taught. We view this relation in terms of a distance measure between models. These notions are demonstrated within the framework of Boolean functions (propositional logic), and some classes of distance measures are characterized. We conclude by suggesting a generic procedure for adaptive diagnosis.

## 1. Introduction and Motivation

While most of this paper is rather abstract and formal, this section is an informal presentation of the ideas that have come to the author's mind while thinking about learning, computer-aided instruction and student modeling. These observations are from the perspective of computer-science, and are not claimed to be based on familiarity with current psychological research. Hence the lack of references to psychological and educational literature within this section. Readers interested in a deeper and detailed discussion of student models are referred to [Clancey 86].

### 1.1 Why Student Modeling?

Student modeling is the process of finding out what the student "really" thinks or knows about the subject matter. The extraction of the student's misconceptions is a crucial component of any effective educational system, regardless of whether it is based on humans or machines.

Teaching may be viewed as a process of transferring knowledge from a knowledge source, the *teacher*, to the *learner*. If the learner is a human, one never knows what has really been transferred. Human students often misunderstand what they have been taught, they may temporarily forget pieces of knowledge, or make casual errors. The internals of the human brain, unlike those of the computer, are not recognized in a sufficient detail, and it is impossible to map pieces of knowledge into their internal representations ("mental images") effectively.

Consequently testing is the common method for evaluating the quality of these obscured mental images. The student's model is judged by his performance in solving problems that require that knowledge. If the student solves a problem correctly, one may assume that his model is functionally correct, at least with respect to the relevant subset of the knowledge. If on the other hand, he or she supplies a wrong answer, the model is possibly incorrect although there may be some other explanations related to human unreliability.

If the problems posed to the student are non-trivial, a correct solution involves several interrelated "chunks" of domain-specific knowledge and problem-solving skills. If one or more of these chunks is missing or incorrect, the student is likely to supply incorrect answers to some or all the questions posed to him. In order to "fix" the student model one must locate the faulty or missing pieces of knowledge that are responsible for the erroneous behavior.

Finding the appropriate explanation for a student's behavior is just another instance of scientific theory formation. Here we have an object demonstrating some input-output behavior, and we want to find an appropriate explanation of this behavior. Not all human teachers have such research capabilities, and none of them can develop individual theories for each of their students due to lack of time and other resources. Thus, the automation of student modeling, if feasible, can have a significant effect on the quality of education.

## 1.2 Student Modeling for Intelligent Tutoring Systems

The paradigm of intelligent tutoring systems (ITS, ICAI), as described in [Clancey 82, Sleeman and Brown 82] has emerged from the knowledge-based system paradigm in AI. Within this framework, knowledge consists of a collection of some symbolic structures that reside in the knower's data-base. According to this paradigm, the mechanization of tutoring activities should be based on an explicit and comprehensible representation of the subject matter. This representation should not only be functionally correct but also be based on a human-oriented formalism. For the moment we are not committed to any particular formalism. The only assumption is that the knowledge-base consists of an appropriate encoding of the subject-matter that enables the computer to solve certain problems in a correct and relatively human-like manner.

Within this framework student modeling may be regarded as a process in which the same problems are posed to the student and to the computerized "expert". If they disagree on some answers, then the the modeling process produces an alternative knowledge-base containing an incorrect or incomplete formulation of the subject matter expressed in a similar formalism. As mentioned earlier, incorrect answers may be casual or systematic. In the latter case the student behaves as having some "bugs" in his version of that knowledge.

Various attempts have been made to mechanize this process of student modeling, e.g, [Young and O'Shea 1981, Sleeman 1983]. One of the relatively successful attempts was the "Buggy" project [Brown and Burton 78, Brown and VanLehn 80, Burton 81] that was concerned with the simple task of arithmetic subtraction. Subtraction has been selected as a representative of procedural skills because it is a purely mechanical algorithm that is not affected by personal experience or beliefs about the world. More than 100 types of systematic errors have been found experimentally

for this skill, although the algorithm can be expressed in less than 10 simple production rules.

As pointed out in [Kawai et al. 86], using the student's answers in order to reconstruct his model is an instance of learning in which the formal teacher "learns" his student from examples. Thus, machine learning techniques [Michalski et al. 83], and inductive inference results [Angluin and Smith 84] are highly relevant to this task. "Real" student modeling is like scientific theory formation in the complicated and weakly-structured domain of psychology. Such a theory should deal with attitudes, intentions, memory, physical conditions, natural language and other topics. Automated student modeling at this level is as far away as is the automation of similar scientific activities in natural domains.

A more modest approach is to neglect all the personal dimensions of the student, and treat him as a neutral entity behaving according to a possibly incorrect or incomplete knowledge-base. This over-simplification is not suggested as a general educational approach, but in certain cases it may be very useful for student modeling because modeling is then reduced to finding a program that explains the student's incorrect behavior. In any case, such simplifications are inherent in most expert-systems.

Even after this simplification, student modeling still looks as a harder task then inferring programs or expert-systems rules. In "conventional" programming, we do not care at all about the formalism and syntax of the program, as long it is semantically equivalent to the program we have in mind. In expert-systems we want the formalism to be comprehensible to humans, and semantic equivalence is insufficient. This requirement is more emphasized in ITS in general and in student modeling in particular. The inferred student model should not only be human-like and consistent with his behavior. It should also reflect his individual choice of formulation. Otherwise, the "bugs" we find in our version of the student's model may be irrelevant to his actual model.

Let us turn now to a formal definition of this simplified student modeling problem. The applicability of results and techniques within this abstract problem to real educational systems is limited to domains that can be naturally formalized in a commonly accepted way, e.g., simple concepts or simple procedural skills.

## 2. Statement of the Problem

The above lengthy verbal discussion can be reduced to the following description:

Suppose a teacher T knows a function $f$ over some domain, i.e., he can calculate its value for all the domain points. T has tried to transfer its knowledge of that function to a learner L. For various reasons to be discussed later, L failed to learn $f$ exactly, but instead he learned another function $g$. T wants to know $g$ while not having any access to L's internal representation of $g$. The only source of information about $g$ are L's answers to queries about the values of $g$ in various domain points.

Inductive inference in general is a very hard task, but in our case there might be some additional information that can make it easier. Knowing that L has been exposed to $f$, one may suspect that $g$ is not an arbitrary function, but a function which is "close" to $f$ in some sense. A similar setting is described in [Shapiro 82] in the context of program debugging: the program to be debugged is an approximation of the correct one. By saying "close" we implicitly assume the existence of a distance measure between functions $d$, such that if $d(g_1, f) > d(g_2, f)$ then, given that $f$ has been taught, $g_2$ is more likely to be learned than $g_1$.

The ability to enumerate functions according to their distance from another function, may give rise to efficient search algorithms for extracting misconceptions. Instead of starting from

an arbitrary function and refining it according to accumulated experience until convergence, the learning algorithm can start from the close function $f$, and explore its neighborhood in increasing-distance order. (The term "distance measure" does not imply that $d$ satisfies all the requirements of a metric; On the contrary, some choices for $d$ are inherently non-symmetric and may not satisfy the triangle inequality).

The correspondence between $d(f,g)$ and the probability of $f$ being misunderstood as $g$ is a desired property, but in order to hold in reality the definition of the distance measure needs to be closely related to the analysis of errors and their origins. Different classes of errors may induce different measures that are very different from each other. A measure $d_1$ may have its maximum at $(f_1, f_2)$ while $d_2$ may be minimal at the very same point.

Beyond the educational application, questions related to distance and similarity among functions and among their representations are interesting for their own sake, and are applicable to domains such as software debugging or hardware fault finding.

As a starting point I have chosen the class of Boolean functions. This is the simplest non-trivial class of functions I know, and many of its relevant properties and applications have been studied from various perspectives (propositional logic, concept acquisition, switching circuits, coding and information theory, pattern classification, image processing, Boolean algebra). Most current rule-based expert-systems do not go beyond propositional reasoning, although this fact may be hidden by syntactic features. For those who are used to other terminologies, a Boolean function is a classification rule that distinguishes between members of two classes according to some yes/no properties.

Negative results within the Boolean framework will usually imply negative results in more complex domains, while positive ones may suggest generalizations to higher classes of functions.

## 3. Boolean Functions and their Representations

In order to investigate the potential classes of errors and their induced measures, let us explore the semantic world of Boolean functions and its relation to some syntactic structures that are used for encoding and communicating such functions.

### 3.1 Boolean Functions and Expressions

An $n$-dimensional Boolean function is a function $f : \{0,1\}^n \rightarrow \{0,1\}$, $B^n \rightarrow B$ for short). A partial Boolean function is a function $f : \{0,1\}^n \rightarrow \{0,1,?\}$, where "?" stands for *undefined* There are $2^{2^n}$ distinct total functions and $3^{2^n}$ partial functions over $B^n$. Any function $f$ over $B^n$ partitions it into three disjoint subsets, $B^+(f)$, $B^-(f)$ and $B^?(f)$, which are the inverse images of 1, 0, and ? respectively.

The most straightforward way to communicate such a function is to list its value at all domain points, i.e., a $2^n$-entry table. We call such an encoding of $f$ as its *semantic* representation.

Boolean expressions (see [Kohavi 78] or other textbooks) provide an alternative way for specifying a Boolean function. A Boolean expression over a set of variables $\{X_1, ..., X_n\}$ is defined inductively as follows:

- Any variable $X_i$ is an expression.
- If $E$ is an expression, so is $(\sim E)$.

- If $E_1$ and $E_2$ are expressions so are $(E_1 \wedge E_2)$ and $(E_1 \vee E_2)$.

An *instance* of a Boolean expression $b$ is determined by assigning values from $\{0,1\}$ to its variables in a consistent manner. By $b(x_1,...,x_n)$ we denote an instance where $x_i \in \{0,1\}$ is assigned to all occurence of $X_i$. The tuple $(x_1,..,x_n)$ is called a *substitution*. The truth-value of an instance, $V(b(x_1,..,x_n))$ is defined inductively:

- If $b$ is 0 or 1 then $V(b) = b$.
- If $b$ is of the form $\sim b_1$, then $V(b) = 1 - V(b_1)$.
- If $b$ is of the form $b_1 \wedge b_2$ then $V(b) = \min(V(b_1), V(b_2))$.
- If $b$ is of the form $b_1 \vee b_2$, then $V(b) = \max(V(b_1), V(b_2))$

The following relations between an expression $b(X_1,...,X_n)$ and an n-dimensional function $f$ are defined:

- $b$ is *positively-consistent* with $f$ if for any substitution
  $V(b(x_1,..,x_n)) = 1 \Rightarrow f(x_1,...,x_n) = 1$.
- $b$ is *negatively-consistent* with $f$ if for any substitution
  $V(b(x_1,..,x_n)) = 0 \Rightarrow f(x_1,...,x_n) = 0$.
- $b$ *represents* $f$ if for any substitution
  $V(b(x_1,..,x_n)) = f(x_1,..,x_n)$.
  $b$ represents $f$ if it is both positively and negatively consistent with i

For example, $X_1 \vee X_2$ is positively-consistent with the following two-dimensional functions:

$$f_1 = (< 0,0,0 >, < 0,1,1 >, < 1,0,1 >, < 1,1,1 >)$$

$$f_2 = (< 0,0,? >, < 0,1,1 >, < 1,0,1 >, < 1,1,1 >)$$

$$f_3 = (< 0,0,1 >, < 0,1,1 >, < 1,0,1 >, < 1,1,1 >)$$

while it represents only $f_1$.

### 3.2 Normal Forms

There are infinitely many expressions that represent a given function. Hence it is worthwhile to define *normal forms* for such expressions:

- A *literal* is a variable or its negation.
- A *monomial* (or a *conjunctive term*) is a conjunction of literals where each variable occurs at most once, e.g., $\sim X_1 \wedge X_2 \wedge X_3$.
- A *clause* (or a *disjunctive term*) is a disjunction of literals where each variable occurs at most once, e.g., $X_1 \vee \sim X_2 \vee X_3$.
- A Boolean expression is in a *disjunctive normal form (DNF)* if it is written as $C_1 \vee .. \vee C_k$, where each $C_i$ is a distinct monomial.
- A Boolean expression is in a *conjunctive normal form (CNF)* if it is written as $D_1 \wedge .. \wedge D_k$, where each $D_i$ is a distinct clause.

An expression is in a *principal DNF* (PDNF) or *principal CNF* (PCNF) if it is a conjunction or disjunction of $n$-literal monomials or clauses. Each term in a PDNF expression corresponds to a function whose $|B^+| = 1$ and the $B^+$ of the function represented by the whole expression is the union of all these points.

Sub-classes of Boolean functions can be defined indirectly by restricting the syntactic forms of their representing expressions. Constraining the syntactic form to $k$-DNF (DNF with terms with up to $k < n$ literals) excludes functions with "too sharp" boundaries between their $B^+$ and $B^-$. In the two-dimensional case, the restriction to 1-DNF, i.e., only disjunctions of literals, prevents the representation of 2 (out of 16) functions, namely the XOR and EQ functions, that require 2-literal conjunctive terms for definition. Other constraints may be imposed on the number of terms ($k$-term DNF), on the sign of the literals (monotone functions) or on the total length of the expression). All such constraints have their counterparts in the semantic domain of the n-dimensional hyper-cube. Usually the loss in expressive power due to such restrictions is accompanied by gains in the cost of storing, computing and learning of functions from such sub-classes.

## 4. Learning Boolean Functions from Examples

The contribution of the knowledge of $f$ and of the distance measure to the rapid convergence to $g$ can be appreciated within the general framework of learning Boolean functions. The general setting of learning is summarized as follows: there exists a black-box such that whenever it is fed with a Boolean tuple, $(x_1, x_2, ..., x_n)$, it outputs a single bit, $f(x_1, x_2, ..., x_n)$. Learning is a process during which we try to identify $f$ from input/output examples. In order to prevent confusion, let us recall that in student modeling the formal "teacher" plays the role of the learning component, and the student is the source of examples for $f$.

### 4.1 Learning as Gradual Covering

The problem of learning a Boolean function from examples is closely related to the known problem from switching theory, namely the synthesis of switching circuits from input-output specifications [Kohavi 78]. This problem can be rephrased as:

Given $B^+(f)$ and $B^-(f)$, find a (usually DNF) Boolean expression $b$ satisfying:

- $b$ is positively-consistent with $f$.
- $b$ is negatively-consistent with $f$.
- $b$ is optimal among other expressions satisfying requirements 1 and 2 with respect to some complexity measure.

This problem is a variant of the extremal discrete covering problem. In such problems we have a domain $D$, a set $R \subseteq 2^D$ and a set $S \subseteq D$. A non-negative cost is assigned to each $r \in R$. The problem is to find a subset $R' = \{r_1, .., r_k\} \subseteq R$ such that each $s \in S$ belongs to at least one $r_i$, and there is no cheaper subset that covers $S$.

The problem of finding a minimal DNF-formula that is consistent with $f$ is an instance of the covering problem where $D = \{0,1\}^n$, the set to be covered is $S_1 = B^+(f)$. $R$ is the set of all $m$-cubes ($m = 1..n$) in $\{0,1\}^n$. In this problem we have an additional constraint: the covering set $R'$ should not cover any element of $B^-(f)$.

The common methods for such a synthesis are the graphically-oriented method of Karnaugh

maps, and its numerical counterpart, the Quine-McLuskey algorithm. This task is easier for "more partial" functions having larger $B^?$, because points from $B^?$ can be annexed to $B^+$ or $B^-$ in order to form "smooth" border lines between both regions. This covering by larger, fewer and consequently cheaper cubes. Unfortunately the problem of finding the optimal cover is NP-complete [Garey and Johnson 79].

The learning problem resembles the synthesis problem with some differences. The first difference is that in learning we do not insist on minimal cost of the synthesized expression. On the other hand we like to exclude trivial expressions (e.g., PDNF) that are nothing but a memorization of the examples observed. So there must be a compromise concerning the optimality of the synthesized expression. The second difference is that in learning we want to infer inductively beyond what can be inferred deductively from the examples. This means that testing the values of $f$ over all the $2^n$ domain points is not our goal. Such an approach removes the essence of learning from the process, and immediately implies exponential complexity.

### 4.2 Approximate Learning

A different approach to this problem has been suggested recently in [Valiant 84, 85] based on the methodology of random sampling. This approach is based on a new definition of learnability of a class of functions. This definition differs from "precise" inductive inference notions (finite identification) in two main aspects:

- The requirement that the algorithm converges to the desired function $f$ is relaxed and replaced by a probabilistic convergence (with probability $1-\delta$) to a function $g$ such that the probability of a domain point to be evaluated differently by $f$ and by $g$ is smaller than $1-\varepsilon$. This requirement is for all functions in the class considered and for all distributions of the domain points.
- A new requirement is added: the whole process must be tractable, i.e., polynomial in $\frac{1}{\delta}$, $\frac{1}{\varepsilon}$ and in $n$.

Once again it is important to note that within this formulation, no prior knowledge about the candidate functions is assumed. This means that all $2^{2^n}$ functions (or a smaller number if we consider sub-classes) are equally likely to be the unknown function. Later we will see why in our model this is not the case.

### 4.3 Construction, Enumeration and Mixed Methods

Inductive inference methods may be divided roughly into constructive (semantics-driven) and enumerative (syntax-driven) methods [Wharton 77, Angluin and Smith 84]. Constructive methods use the observed examples as a basis for the synthesized function. Enumerative methods work on the syntactic domain, starting from an initial expression, sticking to it as long as no new example contradicts it, and moving to the next (according to some order) expression if a contradiction occurs.

Both methods represent a trade-off between two desired properties of the selected hypotheses: *simplicity* and *conformity* with the sample. Enumeration can generate hypotheses in increasing-complexity order so it is guaranteed to find the simplest hypothesis, and it will use the least number of examples [Gold 67]. Constructive methods are eager to cover the current sample, but the price may be a complex representation of the function. This distinction is not sharp, and the properties of the methods depend also on the presentation mode of the examples. The importance of enumerative

methods lies in their completeness, especially in the case of infinite-domain functions. They are, however, computationally exponential by nature. Constructive methods are usually considered more efficient but less general.

In the context of Boolean functions, a trivial constructive method may be to record all examples (which is like maintaining a PDNF representation of the partial function defined by the examples), and from time to time trying to sub-optimize the current expression into a less trivial one. A possible enumeration of DNF formulae may be based on the number of terms, the number of literals and their signs:

$$X_1,\ X_2,\ \sim X_1,\ \ldots\ , X_1 \wedge X_2,\ X_1 \wedge \sim X_2,\ \ldots\ , (X_1 \wedge X_2) \vee (X_1 \wedge \sim X_2) \ldots$$

Boolean functions, as well as other classes, can be partially ordered according to the inclusion relations among their $B^+$ or $B^-$. We say that $f < g$ if $B^+(f) \subset B^+(g)$. (This is the ***subsumption*** relation discussed in [Plotkin 71]). A negative example that contradicts $f$, certainly contradicts $g$. If the functions are represented in a syntactic formalism on which a partial order, consistent with the semantic order, can be defined, then we may substitute the brute-force enumeration procedure with a more adaptive (sample-sensitive) graph searching procedure. For example, if $X_1$ has been contradicted by some negative example then all "larger" expressions such as $X_1 \vee X_2$ can be excluded as well. Refinement methods are applicable to various domains [Laird 86] and they have been used in logic-related domains ,e.g., [Shapiro 82, Mitchell 82]. In the Boolean case, we may start from the "most general" (TRUE) or the "most specific" (FALSE) function, and refine our way up or down the graph according to observed examples. Although the use of partial-order information and refinements may reduce the search space of candidate expressions, the whole process retains its exponential nature. The algorithms suggested in [Valiant 84] are essentially refinement algorithms.

What are the implicit assumptions underlying an enumeration? In the case of simplicity-based order, the assumption is that simpler functions (and simpler syntactic forms) are more probable than complex ones. The simplest expression consistent with the sample will be preferred over other non-contradictory expressions. Thus, the criterion for preferring hypotheses is a mixture of two criteria: a sample-independent simplicity criterion, and a sample-sensitive conformity criterion. This setting can be described within a Bayesian framework [Watanabe 60, Horning 71, Solomomoff 75]:

Given some background knowledge $C$, various hypotheses $H_i$ have a prior (sample-independent) probability denoted by $P(H_i|C)$. For example, if we know that simpler hypotheses are more likely to be true, then our prior probabilities will reflect the simplicity of the hypotheses. The probability that a sample $S_j$ (a collection of examples) will occur given that $H_i$ is true is denoted by $P(S_j|H_i, C)$. In the case of Boolean function with uniform distributions of the examples and no noise, this probability is 0 for all $H_i$ contradicted by $S_j$ and is $kP(S_j|C)$ for the remaining hypotheses. The constant $k$ is the total probability of the rejected hypotheses divided by the number of the unrejected ones. The posterior probability of an hypothesis $H_i$, that reflects both the prior background knowledge and the sample $S_j$ is given by Bayes rule:

$$(*) \quad P(H_i|S_j, C) = \frac{P(H_i|C)P(S_j|H_i, C)}{P(S_j|C)}$$

In the general case the preferred hypothesis is the one that maximizes this posterior probability. As the denominator is hypothesis-independent, we should only maximize $P(H_i|C)P(S_j|H, C)$. Strict maximization of probabilities implies undecidability in infinite domains, and intractability in finite ones. Horning [Horning 71] tackles the first problem in the context of stochastic grammars. An

alternative approach for combining both measures appears in [Gaines 76], where instead of one preferred hypothesis we have a frontier of incomparable hypotheses, each represents a trade-off between the two criteria.

In the deterministic Boolean case (*) implies:

$$P(H_i|S_j,C) = \begin{cases} 0 & \text{if } S_j \text{ contradicts } H_i; \\ kP(H_i|C) & \text{otherwise} \end{cases}$$

So the chosen function will be the most a priori probable function among the uncontradicted functions.

### 4.4 Given vs. Requested Sample

The mode of example presentation may vary in different settings. There may be only positive examples (members of $B^+$), negative examples or both. The domain points can be selected arbitrarily with uniform probability over the domain, under some non-uniform distribution, or requested by the learning algorithm and posed to an oracle who knows the function. Regarding decidability, the order of a presentation is quite immaterial because in the limit, any complete presentation method will produce all the examples that the learner might have asked the oracle about. However when we consider complexity the advantages of "good" examples are significant.

What makes an example better than another one? Each example excludes some functions from consideration and strengthens the probability of the remaining ones. In a stochastic setting functions are not excluded but their probabilities decrease. The utility of an example lies in its discriminatory power, i.e., in its potential influence on the posterior probabilities of the functions in the hypothesis space. This influence can be quantified using some information-theoretic formalism such as expected entropy. The question yielding the maximal expected decrease in entropy should be preferred over all other arbitrary questions. Proceeding this way at each stage, we are likely to converge more rapidly to the desired information state implied by the probabilistic learning criterion.

## 5. Noise and Distance among Boolean Functions

### 5.1 The Origin of Errors

$T$ knows the function $f$ and tries to communicate it to $L$. Let $T_f$ denote the event $\{T$ *taught* $f\}$ and $L_f$ denote $\{L$ *learned* $f\}$. If the communication channels are reliable, the exact knowledge will be transferred, i.e., $P(L_f) = 1$. In the presence of noise of some kind, $P(L_f)$ decreases, and the probability of learning other functions in the neighborhood of $f$ increases. These probabilities depend on two major factors:

- The form of encoding and transmission employed by $T$ during the teaching stage.
- The noise in the channel connecting $T$ and $L$.

It should be noted that noise can appear not only in the channel from $T$ to $L$ but also in other locations. Suppose, for example, $L$ stores its knowledge encoded as a Boolean expression

and answers queries using a noisy interpreter (e.g., one that with some probability exchanges conjunction and disjunction). The observed incorrect behavior will differ in this case from the behavior associated with noise that is present only during the teaching phase. If we recall our initial educational motivation, this is exactly the difference between systematic and casual errors.

The following paragraphs contain an initial discussion of errors and of the probabilities and distance measures associated with them. This discussion is based on two basic methods of knowledge transfer from $T$ to $L$, by examples and by Boolean expressions.

### 5.2 Semantic Distance

The most natural distance measure between two total Boolean functions $f$ and $g$ is the cardinality of the symmetric difference between $B^+(f)$ and $B^+(g)$ or the Hamming distance $d_H$ between the last columns of their corresponding truth tables. This measure is a metric, satisfying:

- $d_H(f,g) = 0 \iff f = g$
- $d_H(f,g) = d_H(g,f)$
- $d_H(f,h) \leq d_H(f,g) + d_H(g,h)$

The values of $d_H$ range from 0 to $2^n$ and when normalized to $[0,1]$ they indicate the error rate of $g$ with respect to $f$. For any function $f$, the number of functions $g$ satisfying $d_H(f,g) = k$ is $\binom{2^n}{k}$. Other variations of this measure may assign different weights to each point, or two different weights for elements from $B^+(f) \setminus B^+(g)$ and $B^+(g) \setminus B^+(f)$.

Weights have been used in [Wharton 74] to extend this measure to infinite-domain functions (languages) by assigning smaller weights to longer strings. Valiant's definition of learnability is expressed in terms of semantic distance: the resulting function is as close as we want to the real one. This is a specialization of Wharton's idea of approximate identification to finite domains.

What kind of a learning protocol may give rise to correlation between the semantic distance $d_H(f,g)$ and $P(L_g|T_f)$? Suppose $f$ has been communicated to $L$ by examples. If the number of examples was less than $2^n$, then $g$ is an approximation of $f$ and $d_H(f,g)$ is a function of the sample and of the learning algorithm. $L$ might have employed a probabilistic learning algorithm based on the value of $f$ at some sample points. A reasonable requirement for such an algorithm is the satisfication of:

$$d_H(g_1,f) > d_H(g_2,f) \iff P(L_{g_1}|T_f) < P(L_{g_2}|T_f)$$

This may not be true for certain samples but the expected probabilities over the sample space should satisfy it. An interesting question here is whether the knowledge of $f$ can contribute to faster convergence of algorithms of the kind described in [Valiant 84, 85], and whether this knowledge can make unlearnable classes learnable (maybe in time polynomial in the distance). Acceleration of convergence seems a consequence of knowing $f$ because the prior distribution of the functions is not uniform and the process starts with a lower entropy.

### 5.3 Syntactic Distance

If $T$ teaches $L$ the function $f$ using Boolean expressions, then a noisy channel may induce other probabilities and distance measure. Suppose, for example, that $f$ is expressed by a monomial.

Monomials over $n$ variables can be encoded as tuples of the form $(s_1, s_2, .., s_n)$ where the values of the $s_i$ denote:

| | |
|---|---|
| $s_i = 1$ | $X_i$ appears in the monomial |
| $s_i = 0$ | $\sim X_i$ appears in the monomial |
| $s_i = ?$ | neither of them appears |

Some examples of this encoding:

| *Code* | *Expression* |
|---|---|
| (001) | $\sim X_1 \wedge \sim X_2 \wedge X_3$ |
| (1?1) | $X_1 \wedge X_3$ |
| (???) | $TRUE$ |

Suppose now that we have a noisy channel with the following properties:

$$P(\{s_1 \ was \ received\}|\{s_2 \ was \ transmitted\}) = \begin{cases} p & \text{if } s_1 = s_2; \\ \frac{(1-p)}{2} & \text{otherwise} \end{cases}$$

where $s_1, s_2 \in \{0, 1, ?\}$ and $p \in (1/3, 1]$.

If the transmitted 2-dimensional function was $f = X_1$ then, assuming independence among the transmissions, the following probabilities are induced:

| *Expressions* | *Probability* |
|---|---|
| $X_1$ | $p^2$ |
| $\sim X_1, TRUE, X_1 \wedge X_2, X_1 \wedge \sim X_2$ | $\frac{p(1-p)}{2}$ |
| $X_2, \sim X_2, \sim X_1 \wedge X_2, \sim X_1 \wedge \sim X_2$ | $\frac{(1-p)^2}{4}$ |

Although the semantic distance $d_H(X_1, \sim X_1) = 4$ is maximal, syntactically $\sim X_1$ is closer to $X_1$ than $X_2$ whose semantic distance from $X_1$ is 2. In this case the noise is symmetric, i.e., $P(s_1|s_2) = P(s_2|s_1)$ and uniform, so a distance measure may be just the number of differences between the two expressions.

Some possible classes of noise are:

- Noise in monomials: deletion of literals, inversion of literals, permutation of variables.
- Noise in DNF expressions: deletion of a monomial, inversion of a monomial (this makes it a disjunctive term), deletion/insertion/inversion of literals in one or more of its monomials.

These generic classes of noise can be further characterized in terms of probabilities and conditional probabilities. A reasonable assumption might be that all the literal-related errors in a DNF expression are located only in a small subset of its monomials.

## 6. Using Distance Information in Adaptive Diagnosis

Suppose $T$ has an ongoing interaction with $L$. Let us assume the existence of an algorithm that can exploit the knowledge of $f$ and of the noise in the channel or in $L$ in order to converge more rapidly to a function near $g$. In this case $T$ may use the following strategy for diagnosing $L$'s misconceptions in learning:

1) Teach $L$ some functions $f_i$.

2) Use an approximate learning algorithm to infer the corresponding $g_i$'s learned by $L$.

3) Find the distance measure (type of noise) that explains the relations among the $f_i$'s and $g_i$'s.

4) Having found the typical noise associated with the student, switch from the general learning algorithm to an algorithm that exploits this information, i.e., an algorithm that specializes on learning close functions.

This "second-order" learning depends on the postulated existence of such specialized algorithms and on the ability to select the measure that explains $L$'s behavior.

Given the definitions of noise and distance, it is possible to generate expressions in a nearest-first order with respect to $f$. This may lead to faster convergence to $g$, while the probabilistic argument (non-uniform a-priori distribution of the function space) can justify the reliance on smaller samples.

The existence of more probable candidate function can be used for selecting queries that distinguish them from each other. For example, the expression $X_1 \wedge X_2 \wedge \sim X_3$ is distinguished from its nearest (under literal deletion errors) neighbors $X_1 \wedge X_2$, $X_1 \wedge \sim X_3$ and $X_2 \wedge \sim X_3$ by the following three points respectively: $< 1,1,1 >$, $< 1,0,0, >$ and $< 0,1,0 >$. Finding good questions for more complex expressions such as DNF is not as straightforward, but it seems to be feasible for some restricted classes of expressions and errors.

If we define functions in terms of other functions, and may query $L$ about values of these sub-functions as well, then some of the techniques used in [Shapiro 82] can be applied. In this case, the background knowledge is about the hierarchical structure of the expression representing the function. In particular, if the representation formalism is k-term DNF, where each term has a meaning, we can learn how $L$ knows each monomial, and then learn his representation of their disjunction. A similar hierarchical approach is suggested in [VanLehn 87].

## 7. Conclusions

In this paper we have discussed student modeling as an instance of machine learning. The implications of the existence of a close model has been studied within the simple class of Boolean functions. It has been formally demonstrated that the ability of the teacher to model the student's behavior is affected by his own understanding of the subject matter, by his acquaintance with the student and by his knowledge of the form in which the subject matter has been taught.

Apparently, there is a big gap between the formalisms required for real-world AI and educational applications, and the simple formalisms within which one can speak using mathematical rigor. We hope that this paper, although it contains neither empirical nor theoretical results, will encourage further efforts towards narrowing this gap.

## ACKNOWLEDGEMENT

I would like to thank Uri Feige, John Gallagher, Muli Safra, Udi Shapiro and Zahava Scherz for their contribution. They constitute a representative sample of the many who were pushed to read various versions of this work during the last year.

## REFERENCES


D. Angluin and C.H. Smith, Inductive Inference: Theory and Methods, *ACM Computing Surveys* 15, 237-269, 1984.

W.J. Clancey, Application-oriented AI research: Education, in E.A. Feigenbaum and A. Barr (Eds.), *The AI Handbook Vol. 2*, 223-294, Los Altos, 1982.

W.J. Clancey, Qualitative Student Models, in J.F. Traub (Ed.), *Annual Review of Computer Science Vol. 1*, 381-450, Annual Review Inc., Palo Alto, 1986.

J.S. Brown and R.B. Burton, Diagnostic Models of Procedural Bugs in Basic Mathematical Skills, *Cognitive Science* 2, 155-192, 1978.

J.S. Brown and K. VanLehn, Repair Theory: A Generative Theory of Bugs in Procedural Skills, *Cognitive Science* 4, 379-476, 1980.

R.B. Burton, DEBUGGY: Diagnosis of Errors in Basic Mathematical Skills, in D.H. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London, 1981.

B.R. Gaines, Behaviour/Structure Transformations under Uncertainty, *Internatinal Journal of Man/Machine Studies* 8, 337-365, 1976.

M.R. Garey and D.S. Johnson, *Computers and Intractability*, Freeman, New-York, 1979.

M.E. Gold, Language Identification in the Limit, *Information and Control* 10, 447-474, 1967.

J.J. Horning, A Procedure for Grammatical Inference, in *Information Processing 71*, North-Holland, Amsterdam, 1972.

K. Kawai, R. Mizogushi, O. Kakusho and J. Toyoda, A Framework for ICAI Systems based on Inductive Inference and Logic Programming, *Proceedings of the 3rd International Conference on Logic Programming*, LNCS 225, 188-202, Springer-Verlag, Berlin, 1986.

Z. Kohavi, *Switching and Finite Automata Theory*, Tata-McGraw-Hill, New-Delhi, 1978.

P.D. Laird, *Inductive Inference by Refinement*, Report DCS-TR-376, Yale University, New Haven, 1986.

R. Michalski, J. Carbonell and T.M. Mitchell (Eds.), *Machine Learning*, Tioga, Palo-Alto, 1983.

T.M. Mitchell, Generalization as Search, *Artificial Intelligence* 18, 203-226, 1982.

G.D. Plotkin, A Further Note on Inductive Generalization, in B. Meltzer and D. Michie (Eds.), *Machine Intelligence 6*, Edinburgh University Press, Edinburgh, 1971.

E.Y. Shapiro, *Algorithmic Program Debugging*, MIT Press, 1982.

D.H. Sleeman, Inferring (Mal) Rules from Pupil's Protocols, *Proceedings of the international Machine Learning Workshop*, 221-227, Monticello, 1983.

R.J. Solomonoff, Inductive Inference Theory - a Unified Approach to Problems in Pattern Recognition and Artificial Intelligence, *Advance Papers of the 4th IJCAI Vol. 2*, 274-280, Tibilisy, 1975.

L.G. Valiant, A Theory of The Learnable, *Communications of the ACM* 27, 1134-1142, 1984.

L.G. Valiant, Learning Disjunctions of Conjunctions, *Proceedings of the 9th IJCAI* Vol. 2, 560-566, Los-Angeles, 1985.

K. VanLehn, Learning One Sub-procedure per Lesson, *Artificial Intelligence* 31, 1-40, 1987.

S. Watanabe, Information-Theoretical Aspects of Inductive and Deductive Inference, *IBM Journal of Research and Development* 4, 208-231, 1960.

R.M. Wharton, Approximate Language Identification, *Information and Control* 26, 236-255, 1974.

R.M. Wharton, Grammar Enumeration and Inference, *Information and Control* 33, 253-272, 1977.

# Graph Induction

G. Deon Oosthuizen

Dept. of Computer Science, University of Strathclyde
26 Richmond Street, Glasgow G1 1XH
deon%uk.ac.strath.cs@arpa.ucl-cs

## Abstract

In this paper we describe Graph Induction[1]. Graph Induction evolved from research regarding the connectionist approach to knowledge representation, i.e. systems in which connections - rather than memory cells - are the principle means of storing information. The method thus adopts a connectionist approach to learning.

Graph Induction is embodied in an algorithm called GRAND (GRAph iNDuction). GRAND is based on a relatively simple knowledge representation model. This makes it easy to integrate with other knowledge-based systems.

Graph Induction is based on a process of continual restructuring of knowledge, occurring as part of update procedures, which results in automatic identification of useful attributes and automatic crystallizing of concepts. It is incremental, and offers two methods to counteract the NP hard problem posed in empirical learning: it employs parallelism - thereby pushing back the computational barriers; as well as delayed inference - adding an implicit approximation capability.

## Introduction

Inductive learning strategies usually involve a search through a space consisting of all possible combinations of the attribute values defining the concepts under consideration or comprising the 'rules' describing groups of objects or events to be classified. Since the basic principles involved in the different learning strategies have been formalized (e.g. [Michalski *et al* 1983]), considerable effort has been put into the problem of economically representing and efficiently searching the potentially huge spaces involved (e.g. [Fischer & Langley 1986], [Ganascia 1987],[Hong & Uhrik 1987]). Graph Induction is another contribution toward achieving that goal. It is based on the view that learning is a memory organisation process [Schank 1982], a "transformation of information"..."into a more efficient and/or more general form in which it is stored for future use" [Michalski 1986].

Graph Induction is interesting because it not only exploits the massively parallel approach offered by connectionism, but it is also based on the familiar concept of tangled hierarchies [Fahlman 1979]. Since class hierarchies often form the basis of representations in knowledge-based systems, it means that the learning process is easy to integrate with the knowledge representation component of such systems. Moreover, Graph Induction occurs as an automatic process; a 'side effect' of update procedures. It is possible to conduct inductive inference only partially in real time, if necessary: obvious inferences can be drawn immediately, and comprehensive, but computationally expensive inferences are postponed until the system is quiescent. The formalism is also useful as a general framework for the description of various learning strategies.

We first describe the method of Graph Induction, followed by some comments and comparisons with related methods. After this, we

[1] Induction by means of graphical representation and graph manipulation.

focus on implementation considerations.

## Graph Induction

The graphical representation on which GRAND is based, is created in the following way. We begin by considering each attribute value as representing a class of objects[2], i.e. the class of all objects having the particular property (value) in common. This is represented by creating a node for the given attribute value and linking it to all objects that are members of the class, by means of arcs pointing to the class. (Each object is also represented by a node.) This procedure is repeated for each value of each attribute recorded in the system, i.e. no pre-knowledge is required to distinguish between useful and irrelevant features. The result is a two-layered graph containing all the objects in the bottom layer and all the attribute values in the top layer. Each object has an upward pointing arc to each of its descriptors (properties) and is thus characterized by the *conjunction* of a given number of features. No arcs are inserted for unknown attribute values.

In essence the GRAND algorithm does the following. The graph structure is restricted in such a way that it adheres to the rule that any two nodes in the graph may have a maximum of one maximal common descendant, i.e. a unique *meet*. This means that a configuration like the one in fig. 1a (containing *multiple meets*) is changed to the one in fig. 1b. It can be shown that this by implication maintains unique minimal common ancestors (or unique *joins)* for any two nodes. Mathematically speaking, it implies the transformation of the graph structure from a *partially ordered set* into a *lattice*. This results in a policy of consistent optimal data factoring and the emergence of a series of tangled class-hierarchies between the original two layers. Because of this obvious relationship with class hierarchies, it is easier to understand the semantics of the network if the nodes are regarded as sets (covering the objects below them in the graph), and the arcs as subset relationships. Unique meets and joins correspond to unique intersections and unions respectively. Transitivity (inheritance) applies as it usually does in such hierarchies.

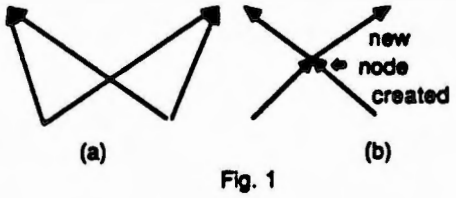



Fig. 1

We now illustrate the operation of the algorithm by means of the following simple example. It contains a high degree of similarity for the purpose of illustration. Consider a number of companies described in terms of the following attributes:

| Attribute | Value |
|---|---|
| type | industrial, financial, mining |
| locality | urban, rural |
| country | France, Germany, Japan, UK, USA |
| ownership | private, public |

Suppose the following examples are volunteered to the system.

| Example | Description | | | |
|---|---|---|---|---|
| cpy_1 | financial | urban | USA | private |
| cpy_2 | financial | urban | UK | public |
| cpy_3 | financial | urban | UK | private |
| cpy_4 | financial | urban | UK | public |

Fig. 2a shows the (partial) GRAND network after the first two examples have been inserted. Fig. 2b shows how fig. 2a is transformed (to eliminate the multiple meets at CPY_1 and CPY_2) to form a new node R, and how the third example is added to the structure. This triggers another transformation (to eliminate the multiple meets at CPY_2 and CPY_3) resulting in the creation of node S (see fig. 2c). Additionally, the multiple meets of R and the

[2] We use the term *object* to refer to the entities which are observed by and represented in the system, but could equally well have chosen terms like *event* or *pattern*. The only assumption is that these complex entities can be described in terms of vectors of primitive features or attribute values.

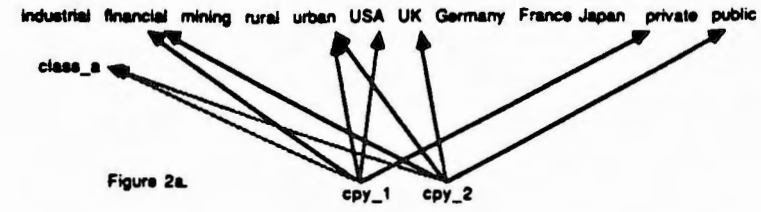



Figure 2a.

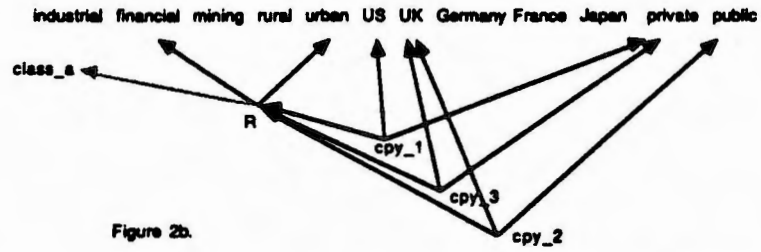



Figure 2b.

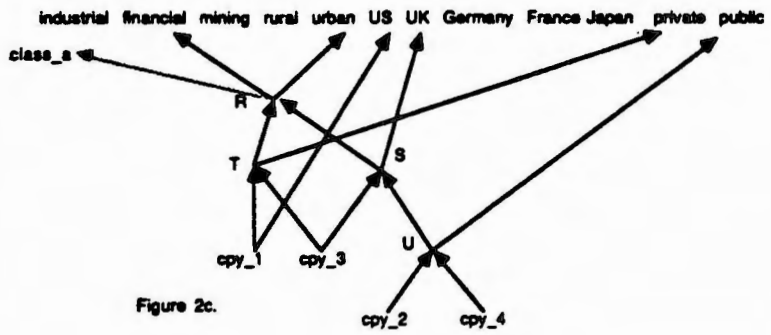



Figure 2c.

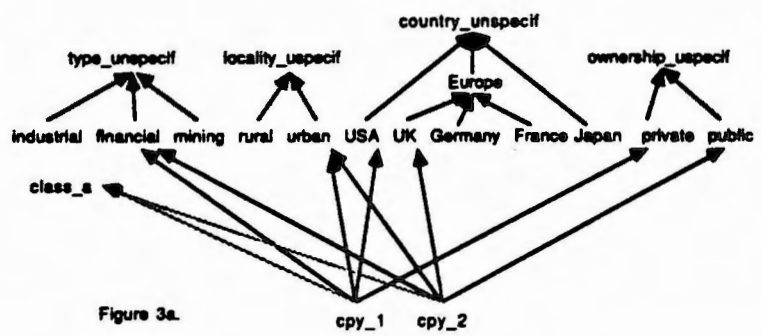



Figure 3a.

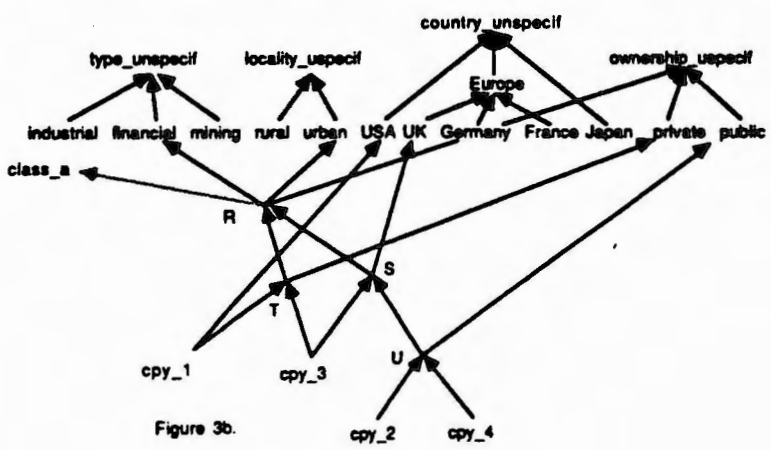



Figure 3b.

node PRIVATE at CPY_1 and CPY_2 are removed, resulting in the creation of node T (see fig. 2c). The insertion of CPY_4 causes the creation of node U.

The significance of the GRAND algorithm is that it maintains a regime of maximal concept integration. Maximally common concepts[3] (classes) are identified and objects are clustered together accordingly. The application of the transformation rule corresponds to the generalization rules described by Michalski in his theory of inductive learning [Michalski 1983]. The generalization rules are invoked implicitly during the transformation of different variants of semantic network configurations. Thus, the application of above transformation to the network structure provides inductive learning. As changes occur in the universe of discourse, nodes and arcs are constantly added and removed, transformations are continuously triggered, and new concepts are formed. Consequently, we obtain what could be described as a "self-learning" network.

The reason why this takes place is that the unique meets/joins referred to above, correspond semantically to *maximally specific concepts* - which are corner-stones of the inductive process. The 'class' hierarchy created provides a sequence of concepts of increasing generality (upward) or specificity (downward) for each object in the graph. Because these sequences are optimally integrated, such a sequence exists for every possible group/combination of objects or lower order concepts in the graph. The path between the maximally specific concept of any object set and each of its potential *maximally general discriminant concepts* is also contained in the graph.

In fig. 2c, node S constitutes a maximally specific concept for the object set

O = {CPY_2,CPY_3,CPY_4}.

The concept S is by definition characterized by the conjunction of the attribute values 'above' it in the graph, i.e. (FINANCIAL, URBAN, UK).

[3] We use the term *concept* to refer to the crystallized classes - each being a combination of features that manifest together and as such serves as a building block for the description of other concepts. It can be regarded as a name associated with a characteristic description.

GRAND is extremely sensitive to the existence of these concepts and it consistently caters for their timely creation. Thus, we obtain an automated mechanism for learning from observation.

It can be shown [Oosthuizen 1986] that the generalized descriptions thus derived, correspond to those created by Lebowitz' "similarity-based generalization" system, UNIMEM [Lebowitz 1986]. However, in UNIMEM, an object can be stored in several places in memory, because concept hierarchies are not integrated as in GRAND. Also, the representation of all attributes as structured variables in GRAND, enables an additional level of generalization. To improve the generalization power of the induction mechanism, all attributes - including nominal and ordinal attribute types - are represented as structured types. This is done by linking nominal attribute values (e.g. blue,red) to a common superclass (e.g. any_colour or colour_unspecif). As a result, GRAND is not able to handle attribute types having a very large number of values (like numerical types with unrestricted ranges) unless these can be grouped together in some or other way, like using intervals containing subranges of numerical values. Intermediate collective nodes may be inserted as required by the system. In the above example, FRANCE, GERMANY and the UK are linked to a node labeled EUROPE. This enables the system to generalize to the degree necessary.

This extension has some implications for the above example. Within the enlarged context, new *multiple meets* appear and the graph becomes liable for an additional transformation when CPY_2 is added. The configuration of fig. 2a now appears as shown in fig. 3a. The nodes FINANCIAL, URBAN and OWNERSHIP_UNSPECIF have multiple meets at CPY_1 and CPY_2. The new final configuration is shown in fig. 3b.

To prevent over-generalization, we delay transformation in the following situations. In cases where more than two attribute values are involved in a generalization, transformation is inhibited until all of the attribute values can be incorporated in one transformation. E.g. in the case of company type, three mutual meets between TYPE_UNSPECIF and another node are

required (incorporating each one of the individual values INDUSTRIAL, FINANCIAL and MINING) before a transformation is performed.

GRAND can equally well serve as the basis for **learning from examples** by explicitly assigning each object to a predetermined class. In that context, the unique meets and joins provide most specific or *maximal characteristic descriptions*. Assuming that all the objects in the above example belong to the same class, this can be done by inserting the grey arcs as indicated in fig. 2a. The grey arcs are affected by the transformations as indicated by fig. 2b-c and eventually link the predetermined class-node to one or more other nodes (concepts). Each of these concepts is defined by the attribute values to which it is connected and thus constitutes a characteristic description for the class involved. A link to a single concept indicates the presence of a maximally specific conjunctive generalization. Multiple links indicates disjunctive concepts, i.e. a 'rule' containing disjunction.

From fig. 3b we can derive the rule:

if type=FINANCIAL AND locality=URBAN
AND ownership=OWNERSHIP_UNSPECIF

then class=CLASS_A

The last term in the condition part reflects a 'don't care' value and can therefore be dropped.

Since an attribute value can be considered as a class on its own (as mentioned before), the same mechanism can be used to determine dependency relationships between any particular attribute value and all the other attribute values or concepts related to it.

## Discussion

Graph Induction is an incremental learning process, occurring automatically during the course of routine updating procedures. Adding an object corresponds to inserting a number of arcs (see fig. 2a), each of which could trigger a transformation. Even if only one new attribute value of an object becomes known and is added to the knowledge base, the single new arc involved may trigger a transformation. Thus 'learning' starts from the least bit of available information and inferences are constantly updated in an optimal way and at the earliest possible stage[4], taking into account the most recently added data. It therefore occurs as an autonomous process.

These autonomous activities (transformations) may occur off-line, i.e. not in real time. In other words, they can be considered as background activities proceeding during the normal course of operation of the system but also when the system is not in use.

The Graph Induction formalism is based on the familiar concept of class hierarchies. Each newly-formed concept is linked either directly to its descriptors or to the concepts in terms of which it is defined. The acquired knowledge is therefore stored in a form transparent and easily understandable to the user. This compares favourably with, for example, the use of a more abstract structure like the Extension Matrix employed in [Hong & Uhrik 1987].

Since the conjunction of maximally specific concepts (or characteristic descriptions) comprises rules, a decision tree can be derived from the graph at any stage. However, the graph structure itself forms an optimal indexing mechanism and is an excellent vehicle for data retrieval (see later section on connectionism). It is *not* necessary to know the useful attributes beforehand as in some other systems, e.g. ID3 [Quinlan 1983]. In GRAND they are identified automatically.

## Related Work

Data factoring ('detection of similarities' or 'finding longest one-to-one matches') forms the basis of various empirical learning methods proposed during the past 10-15 years [Dietterich & Michalski 1983]. In terms of similarity to GRAND, Fu & Buchanan [1985] went as far as actually formulating a heuristic rule for Naming Switchover Points, stating

[4] Subject to the degree to which transformations are performed in real time, as we shall explain in a moment.

if i) There are subsets of n low level nodes and m high level nodes such that *all* n lower level nodes have (confirming) links to *all* m high level nodes,
ii) n>1 and m>1 and mn>4

then it is worthwhile to define a common intermediate node.

By removing the 'mn>4' restriction, we expose ourselves to crowding the network. On the other hand, we gain the strong characteristics of the lattice structure which, as Ganascia [1987] points out, is:
(1) *complete* in the sense that *all* logical relations between features/concepts are captured
(2) *minimal* in the sense that each feature/concept appears maximally once in the representation. This accounts for the prominence of *connections*.

Furthermore, embedding the process in a wholly transitive structure enables us to detect similarities across levels of generality, or implied by higher levels of generality.

Ganascia describes a structure called a *Hilbert cube* which forms the basis for learning in the CHARADE system [Ganascia 1987]. A Hilbert cube comprises a hypercube structure also possessing the properties of a Boolean lattice. An *n* dimensional Hilbert cube is created, where *n* is the number of individuals contained in the learning set. Each of the *n* axes of the cube, numbered from 1 to *n*, is associated with an object. Each vertex corresponds to a set of objects - set membership being defined by the projection of the vertex onto the corresponding axis. Thus, each vertex is labeled by its cartesian coordinates, where a 1 or 0 in position *i* indicates the presence or absence (resp.) of object *i* in the corresponding set. A similar cube is created for the set of features ('descriptors'). Functions are defined which map the cubes onto each other and remove redundancies (refer to [Ganascia 1987] for further detail).

As in the case of Graph Induction, objects and features are clustered together in sets which are related by means of inheritance relationships in structures that form lattices. Therefore, disregarding null vertices, the combined Hilbert cubes of descriptors and of objects constitute a formalism that is structurally and semantically equivalent to the GRAND network. GRAND differs from CHARADE in the way the structure is represented and manipulated.

Ganascia describes a generalization technique by means of which it is possible to generalize atomic descriptors to intervals of values. He also describes statistical rules which can be used to derive approximate rules, expressed in probabilistic terms, from the graph. These can be used to identify 'outliers' and alter the graph accordingly - a means of evading the ubiquitous problem of *noise* present in the data.

Lebowitz [1986] uses each new object to check whether current 'generalizations' are confirmed or contradicted and thus maintains a confidence level for each introduced concept. Such evidence is accumulated until a pre-specified threshold is exceeded, when the concept is 'frozen' and considered valid.

This method can be used to prevent the network crowding referred to above. Contradicted concepts can be deleted. Additionally, nodes which have 'building block' status only, may under certain conditions be merged with other nodes. (These are nodes which have not been explicitly named by the user.)

An alternative way to approach this 'approximate' induction, is via interaction with a human operator, i.e. by resorting to oracle directed induction. This process is closely related to the work on hypothesis formation done by Elgalal [1985]. The information contained in the induction graph is used as a basis for a man-machine dialogue in which eminent features in the available data are highlighted for the human operator in the form of hypothetical concepts. This opens the door to the structuring of knowledge by asking questions. Muggleton's Duce system [Muggleton 1987] exemplifies the possibilities of this method, using inductive operators that have a similar effect as the GRAND transformations.

The completeness of logical relations referred to earlier on has some implications in the areas of data analysis and conceptual schema design. Because of the prominent role of functional dependencies in database theory, such

a complete inventory of logical relations could be of great value in establishing dependencies between attributes.

Siebes & Kersten [1987] introduced a new formal model for the description and analysis of database semantics that has surprisingly much in common with the representation formalism described here. They show that the database intension can be cast in a topological space constructed out of attributes. From this they can derive the extension through well-defined mappings.

*SUPERGRAN*

GRAND has been incorporated in a system called SUPERGRANd [Oosthuizen 1987]. SUPERGRAN exploits the interaction between two learning methods: a genetic search mechanism 'discovers' useful strings of attribute values and thereby creates sample events which are assimilated employing the concept acquisition and generalization capabilities of GRAND. "Champions" are identified among features or combinations of features and used to advise the genetic discovery mechanism on where to look for "better" strings. Thus, a higher degree of coherence is established between the event of a significant discovery, or underlying trend, and the present exploration of the search space.

**Implementation Considerations**

**a. Complexity**

Although the basic idea behind Graph Induction - the restructuring of a graph into a "canonical" form - is quite straightforward, its application poses a computational problem. This is illustrated in fig. 4. Let us say a certain schema A is described by 6 features (e-j) and another schema B is added to the graph. According to the graph induction rule, fig. 4 has to be transformed to the configuration in fig. 5. But, recognizing multiple meets/intersections, like those at A and B, requires the inspection of the conjunction of all possible combinations of pairs of sets intersected by B. If we want to arrive at the configuration in fig. 5, it requires the inspection of all combinations of all higher order conjunctions as well, i.e. groups of 3 sets, 4 sets and so on must be examined. Thus, there is an exponential increase in the number of conjunctions to be inspected as the number of features of B increases. To make things worse, sets may have 'semi-explicit' intersections. Although the sets themselves do not intersect, their indirect descendents may well do so (see fig. 6). Consequently, the inspection has to be conducted to depths of greater than one level.

In order to link each of $n$ objects to $m$ features, $n*m$ arcs are required. It can be shown that the application of a transformation generally reduces this number of links to $n+m$, and increases the number of nodes in the graph by one. In the case of above semi-explicit intersections, however, some configurations imply the insertion of an arc and no extra nodes. In the worst case, four arcs are added, as well as a node. The reader may verify this by substituting two or more arcs for every single arc in fig. 1a, in which case transformation implies

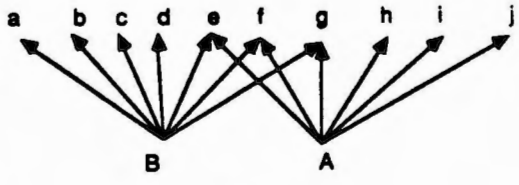



Fig. 4

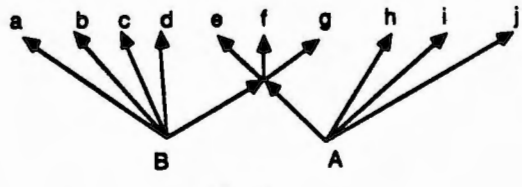



Fig. 5

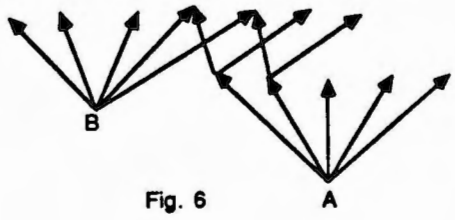



Fig. 6

superimposing the configuration of fig. 1b on the new version of fig. 1a.

Preliminary tests suggest that, on average, the overall number of links remains roughly constant and the number of nodes tends to double. This is illustrated by fig. 3b, where the number of links changed from an initial 16 to 15, and 4 new nodes were created (ignoring feature nodes and links between them as well as the grey arcs).

Naturally, these statistics are data dependent. Still, they are exceptable, especially in the light of the availability of the special purpose associative processing hardware, to be described in the next paragraph.

**b. Graph Induction and Connectionism**

As mentioned earlier, Graph Induction originated from a study of the knowledge representation capabilities of massively parallel connectionist networks.

A set processing machine has been developed [McGregor *et al* 1987] which is able to operate on vast quantities of data items. Subsequently software as well as hardware implementations (prototypes) have been created as part of the low level functions of the FACT Expert Database System. The hardware version of the system, known as Generic Associative Memory (GAM), dynamically constructs networks to process set data in a massively parallel fashion which makes processing speed virtually independent of the width of the network. These systems are described in detail elsewhere [McGregor *et al* 1987].

The core operation of this set processing system, involves the *closure* of a set (node). The *upward closure* of a node includes the node itself as well as all nodes 'above' it in the network, i.e. following the arrows. The *downward closure* includes all nodes below the node of closure. The determination of the intersection of closures is implemented as a hardware function.

As mentioned above, transformations are triggered by the detection of multiple intersections. The GAM closure operation is ideal for the detection of multiple intersections and especially semi-explicit multiple intersections in complex networks (i.e. where the multiple intersections exist at levels lower than the immediate descendants of the two nodes involved).

The use of the GAM closure operation in principle eliminates the distinction between explicit and semi-explicit intersections. Because of its inherent parallelism, the closure operation is also independent of the width of the "tree" involved - i.e. whether a particular node has 2 or n descendants (where n >> 2). Furthermore, the intersection of closures of two nodes like A and B in fig. 4 is (1) a primitive operation in GRAND and (2) executed in a minimal amount of time. GAM ensures the feasibility of the application of Graph Induction as a general purpose learning method.

The prototype NMOS GAM devices developed previously deliver complete closures in the oder of 10 microseconds on up to 16k nodes. The current version (in CMOS technology) has a design target of 100k nodes.

Even the software version of the above set processing machine has been refined to a practical alternative in its own right. It is based on invertable bit string representations and employs memory caching. During 4 hour trials on 10-20k nodes on a Sun-3 computer, it took on average 1 millisecond to perform individual closures. However, this figure is highly dependent on the tangledness of trees, i.e. the amount of overlap occurring in subnetworks, and in the worst case it took up to half a second to complete one closure.

*Delayed Induction*

Apart from the use of special hardware, an additional strategy can be adopted to ease the severity of the computational task incurred by the 'brute force' nature of Graph Induction.

By close inspection one finds that an exhaustive coverage of all nodes which may be transformed requires visitation of every pair in the graph whenever an arc is added to the network. However, it seems to be the case that, most of the time transformations are triggered by nodes near the arc involved. It is difficult to

explain what we mean by *near the arc*, though. Apparently, most of the eligible transformations are performed, even though search is restricted to the immediate parents and children of the nodes to be connected by the arc in question. This observation has not yet been fully researched, though, and is therefore quite speculative. However, it seems reasonable to expect that it might be possible to find an appropriate strategy allowing a minimum of restricted online induction and postponing exhaustive (or at least extensive) coverage of the search space until the system is quiescent.

An assumption we make regarding the practical implications of this differentiation, is that the distance between the nodes involved in a transformation is related to the degree to which the inference could intuitively be considered as relevant or apparent. (Note that this *distance* is just a vertical distance, i.e. the difference in levels of generality of the nodes partaking in a transformation. It is not to be confused with distance in the context of explicit models of human associative memory (Cf. [Narayanan 1986]).)

## Conclusion

Graph Induction supports induction of knowledge from examples and from observation. Integrated with the knowledge representation component of a knowledge-based system, it becomes an autonomous mechanism, sustaining knowledge acquisition and refinement. It also serves as a basis for interactive knowledge acquisition methods. Graph Induction goes far along the path of integrating three very basic aspects often mentioned in connection with learning: generalization, search and indexing. It employs parallelism and allows differentiated and delayed inference - both principles evident in human reasoning and cognition.

Graph complexity remains an area of concern. We believe that the solution lies along the line of integrating Graph Induction with explanation-based learning methods, i.e. in the use of domain knowledge to restrict or prune the network, rather than other more artificial mechanisms.

## Acknowledgements

I am indebted to my supervisor, prof. Douglas McGregor, to dr. Stephen McInnes for proof reading this paper, and to Jesus Christ, the Origin of all knowledge.

## References

Dietterich, T.G., Michalski, R.S. [1983] A comparative review of selected methods for learning from examples. In Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (Eds.) Machine learning: An Artificial Intelligence Approach, Tioga Publishing Company, Palo Alto, CA.

Elgalal, E.M.A. [1985] Minimally-redundant data structures and reasonable hypotheses: some general heuristic methods of knowledge processing. Ph.D. Thesis. University of Strathclyde, Glasgow.

Fahlman, S.E. [1979] NETL: A System for Representing and Using Real World Knowledge. The MIT Press, Cambridge, MA.

Fischer, D., Langley P. [1986] Conceptual Clustering and Its Relation to Numerical Taxonomy. In Artificial Intelligence and Statistics, W.A. Gale (Ed.). Addison-Wesley.

Fu, L.M., Buchanan, B.G. [1985] Learning intermediate concepts in constructing a hierarchical knowledge base. Procs. of the 9th International Joint Conference on Artificial Intelligence. Los Altos, CA: Kaufmann, pp 659-666.

Ganascia, J-G. [1987] Learning with Hilbert Cubes. In Progress in Machine Learning, I. Bratko, N. Lavrac (Eds.). Sigma Press, Wilmslow.

Hong, J., Uhrik, C. [1987] The Extension Matrix Approach to Attribute Based Learning. In Progress in Machine Learning, I. Bratko, N. Lavrac (Eds.). Sigma Press, Wilmslow.

Lebowitz, M. [1986] UNIMEM, a general learning system: An overview. Proceedings of

ECAI-86, Brighton, England.

McGregor, D.R., McInnes, S.T., Henning, M. [1987] An Architecture for Associative Processing of Large Knowledge Bases. To appear, The Computer Journal.

Michalski, R.S. [1983] A Theory and Methodology of Inductive Learning. In Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (Eds.) Machine learning: An Artificial Intelligence Approach,. Tioga Publishing Company, Palo Alto, CA.

Michalski, R.S. [1986] Machine learning research in the artificial intelligence laboratory at Illinois. In T.M. Mitchell, J.G. Carbonell and R.S. Michalski (Eds.) Machine Learning: A guide to Current Research. Kluwer Academic Publishers.

Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (Eds.) [1983] Machine learning: An Artificial Intelligence Approach, Tioga Publishing Company, Palo Alto, CA.

Muggleton, S. [1987] Structuring Knowledge by Asking Questions. In Progress in Machine Learning, I. Bratko, N. Lavrac (Eds.). Sigma Press, Wilmslow.

Narayanan, A. [1986] Memory models of man and machine. In M. Yazdani (Ed.) Artificial Intelligence: principles and applications. Chapman and Hall.

Oosthuizen, G.D. [1986] A Paradigm for Automatic Learning. Internal Report FACT 24/86. University of Strathclyde, Scotland.

Oosthuizen, G.D. [1987] SUPERGRAN: a connectionist approach to learning integrating genetic algorithms and graph induction. Procs. of 2nd International Conference on Genetic Algorithms and their Applications. MIT, Cambridge MA, July 1987.

Quinlan, J.R. [1983] Learning efficient classification procedures. In Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (Eds.) Machine learning: An Artificial Intelligence Approach, Tioga Publishing Company, Palo Alto, CA.

Siebes, A., Kersten, M.L. [1987] Using Design Axioms and Topology to Model Database Semantics. Procs. of the 13th International Conference on Very Large Data Bases, Brighton, September 1987.

# INDEX TO AUTHORS

*T R Addis* **A1**
Computer Science Dept., University of Reading, Whiteknights, PO Box 220, READING, Berks RG6 2AX

*A Anjewierden* **E2**
Dept. of Social Science, Informatics, University of Amsterdam, Herengracht 196, 1016 BS Amsterdam, THE NETHERLANDS.

*Jill Bell* **D2**
JB Associates, 47 Western Road, WINCHESTER, Hants, SO22 5AH.

*R Blumenthal* **F1**
Marconi Research Centre, West Hanningfield Road, Great Baddow, CHELMSFORD, Essex CM2 8HN

*J Boose* **A6**
Knowledge Systems Lab., Boeing Advanced Technology Center, 7L-64, Boeing Computer Services, PO Box 24346, Seattle, Washington 98124, USA

*J M Bradshaw* **F2**
Computer Science Dept., Thames Polytechnic, Wellington St., LONDON SE18 6PF

*J Breuker* **B1**
Dept. of Social Science Informatics, University of Amsterdam, Herengracht 196, 1016 BS Amsterdam, THE NETHERLANDS

*A M Burton* **D3**
Dept. of Psychology, University of Nottingham, NOTTINGHAM NG7 2RD

*B Clarke* **C2**
AI Group, Cambridge Consultants, Science Park, Milton Road, CAMBRIDGE CB4 4DW

*C N Cooper* **C3**
Dept. of Sociology, University of Surrey, GUILDFORD, Surrey GU2 5XH

*P Corsie* **F1**
Marconi Research Centre, West Hanningfield Road, Great Baddow, CHELMSFORD, Essex CM2 8HN

*L A Cox* **E3**
US West Advanced Technologies, 6200 S. Quebec St., Englewood, CO. 80011, USA

*D Diaper* **D4**
Psychology Section, Dept. of Combined Studies, Liverpool Polytechnic, C F Mott Campus, Liverpool Road, Prescot, Merseyside L34 1NP

*J Diederich* **E4**
Gesellschaft fur Mathematik und Datenverabeitung, GMD, Postfach 1240, Schloss Birlinghoven, D-5205 Sankt Augustin 1, WEST GERMANY

*L Esfahani* **E5**
IT Research Faculty, Electrical Engineering Dept., Brighton Polytechnic, Cockroft Building, Lewis Road, BRIGHTON BS2 4GJ

*Brian R Gaines* **A3 B2 C1**
Computer Science Dept., University of Calgary, Calgary, Alberta, CANADA T2N 1N4

*J Gammack* **F3**
California Inst. of Technology, Electrical Engineering, 116-81 Pasadena, CA 91125, USA

*R J Hardiman* **E6**
Education Centre, IBM, Home Farm, Leigh Road, EASTLEIGH, Hants SO5 4EU

*H L Hausen* **D5**
GMD, Schloss Birlinghoven, D-5205 Sankt Augustin 1, FEDERAL REPUBLIC OF GERMANY

*A P Hedgecock* **A2**
Computer Science Dept., Brunel University, UXBRIDGE, Middx UB8 3PH

*Y Kodratoff* **A5**
Inference & Learning Group, LRI, Universite Paris-Sud & CNRS, Bat. 490-91405, Orsay, FRANCE

*O Laske* **D8**
Computer Science Dept., Boston College, Chestnut Hill, Massachusetts 02167, USA

*Marc Linster* **D1 E4**
Gesellschaft fur Mathematik und Datenverabeitung, GMD, Postfach 1240, Schloss Birlinghoven, D-5205 Sankt Augustin 1, WEST GERMANY

*O Maler* **F4**
Weizmann Inst., Rehovot, ISRAEL

*B McHugh* **C4**
Inst. for Nuclear Technology, Chalmers University of Technology, 4126 Gothenburg, Sweden (TEL. 010 46 31721432)

*I Monarch* **E1**
Computer Science Dept., Carnegie-Melon university, Pittsburgh, PA 15213, USA

*K Morik* **A4**
Technische Universitat Berlin, Institut fur Angewandte Informatik, Projekt KIT-Lerner, Sekr FR 5-8, Franklinstr 28/29, D-1000, Berlin 10, WEST GERMANY

*D Needham* **F1**
Marconi Research Centre, West Hanningfield Road, Great Baddow, CHELMSFORD, Essex CM2 8HN

*H J Neusser* **D5**
GMD, Schloss Birlinghoven, D-5205 Sankt Augustin 1, FEDERAL REPUBLIC OF GERMANY

*S Nirenburg* **E1**
Computer Science Dept., Carnegie-Melon university, Pittsburgh, PA 15213, USA

*D E O'Leary* **D6**
Graduate School of Business, Univ. of Southern California, Los Angeles, California 90089-1421, USA

*Michael D Oliff* **F5**
Dept. of Computer Science, University of Strathclyde, 26 Richmond St., GLASGOW G1 1XH

*G Rugg* **D3**
Dept. of Psychology, University of Nottingham, NOTTINGHAM NG7 2RD

*I Ruhmann* **E4**
Gesellschaft fur Mathematik und Datenverabeitung, GMD, Postfach 1240, Schloss Birlinghoven, D-5205 Sankt Augustin 1, WEST GERMANY

*N R Shadbolt* **D3**
Dept. of Psychology, University of Nottingham, NOTTINGHAM NG7 2RD

*Maurice Sharp* **C1**
Computer Science Dept., University of Calgary, Calgary, Alberta, CANADA T2N 1N4

*P Smyth* **F3**
California Inst. of Technology, Electrical Engineering, 116-81 Pasadena, CA 91125, USA

*F N Teskey* **E7**
Dept. of Psychology, University of Warwick, COVENTRY CV4 7AL

*S Thorpe* **E8**
Inference & Learning Group, LRI, Universite Paris-Sud & CNRS, Bat. 490-91405, Orsay, FRANCE

*G Trimble* **C5**
Dept. of Civil Engineering, University of Technology, LOUGHBOROUGH, Leics LE11 3TU

*T Uthmann* **D6**
Graduate School of Business, Univ. of Southern California, Los Angeles, California 90089-1421, USA

*M Welbank* **C6**
RM 218, 14 Finsbury Circus, LONDON EC2M 7DP

*B Wielinga* **B1**
Dept. of Social Science Informatics, University of Amsterdam, Herengracht 196, 1016 BS Amsterdam, THE NETHERLANDS

*R V Williams* **E7**
Dept. of Psychology, University of Warwick, COVENTRY CV4 7AL

*R M Young* **D7**
MRC Applied Psychology Unit, 15 Chaucer Road, CAMBRIDGE CB2 2EF