

GMD-Studien Nr. 143

John Boose, Brian Gaines,
Marc Linster (EDS.)

Proceedings of the European Knowledge Acquisition Workshop (EKAW'88)

June 19–23, 1988

June 1988

GESELLSCHAFT FÜR MATHEMATIK
UND DATENVERARBEITUNG MBH



GMD-Studien Nr. 143

John Boose, Brian Gaines,
Marc Linster (EDS.)

Proceedings of the European Knowledge Acquisition Workshop (EKAW'88)

June 19–23, 1988

June 1988

GESELLSCHAFT FÜR MATHEMATIK
UND DATENVERARBEITUNG MBH

© GMD 1988

Alle Rechte vorbehalten. Insbesondere ist die Überführung in maschinenlesbare Form sowie das Speichern in Informationssystemen, auch auszugsweise, nur mit schriftlicher Einwilligung der GMD gestattet.

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the GMD.

Anschriften der Herausgeber / Addresses of the editors:

John Boose
Boeing Artificial Intelligence Centre
Boeing Computer Services M/S 7A-03
PO Box 24 346
Seattle, Washington, USA 98124

Brian Gaines
Department of Computer Science
University of Calgary, Calgary
Alberta, Canada T2N 1N4

Marc Linster
Institut für Angewandte Informationstechnik
der Gesellschaft für Mathematik
und Datenverarbeitung mbH
Schloß Birlinghoven
Postfach 1240
D–5205 Sankt Augustin 1

CIP-Titelaufnahme der Deutschen Bibliothek

European Knowledge Acquisition Workshop <1988, Bonn>:
Proceedings of the European Knowledge Acquisition Workshop
: June 19 – 23, 1988 / John Boose ... Ges. für Mathematik u.
Datenverarbeitung mbH. – Sankt Augustin : GMD, 1988
(GMD-Studien ; Nr. 143)
Auf d. Haupttitels. auch: EKAW '88
ISBN 3–88457–143–5
NE: Boose, John [Hrsg.]; Gesellschaft für Mathematik und
Datenverarbeitung <Sankt Augustin>: GMD-Studien

ISSN 0170–8120

ISBN 3–88457–143–5

Gesellschaft für Mathematik und Datenverarbeitung mbH

Schloß Birlinghoven
Postfach 1240
D-5205 Sankt Augustin 1

Telefon (02241) 14–1
Telex 8 89 469 gmd d
Telefax (02241) 14 28 89
Teletex 2627-224135=GMDVV

Proceedings of the European Knowledge Acquisition Workshop for Knowledge-Based Systems

EKAW88

EKAW88 (European Knowledge Acquisition Workshop) held in Bonn, Germany from June 19th till June 23rd 1988 is the second European Workshop concerned with knowledge acquisition for knowledge based systems. The first one was held in Reading, England in 1987. It was organized by Tom Addis, John Boose and Brian Gaines. Other workshops in this area took place in Banff, Canada in 1986 and 1987.
It is the objective of EKAW88 to assemble theoreticians and practitioners of AI who recognize the need for developing methods and systems to assist the process of acquiring and modeling knowledge for knowledge-based systems.
Topics of EKAW88 are :

o Transfer/modeling of expertise - systems that obtain and model knowledge from experts.
o Transfer/modeling of expertise - manual knowledge acquisition methods and techniques.
o Apprenticeship, explanation-based, and other learning systems; integration of such systems with other knowledge acquisition techniques.
o Issues in cognition and expertise that affect the knowledge acquisition process.
o Extracting and modeling of knowledge from text. o Integration of knowledge acquisition techniques within a single system; integration of knowledge acquisition systems with other systems (hypermedia, database management systems, simulators, spreadsheets...).
o Knowledge acquisition methodology and training.
o Validation of knowledge acquisition techniques; the role of knowledge acquisition techniques in validating knowledge-based systems.

Workshop Co-chairmen:

John Boose, Boeing Computer Services
Brian Gaines, University of Calgary
Marc Linster, GMD

Program Committee:

Tom Addis, University of Reading
Jeffrey Bradshaw, Boeing Computer Services
Yves Kodratoff, Université Paris-Sud
John McDermott, Digital Equipment Corporation
Ryszard Michalski, University of Illinois
Katharina Morik, TU Berlin
Wolfgang Schoenpflug, FU Berlin
Mildred Shaw, University of Calgary
Bob Wielinga, University of Amsterdam
Marion Wittstock, AWFI Berlin

The workshop is financially and organisationally supported by the GMD.

Table of Contents

Open day :

" A concept language for knowledge elicitation" 1-1
T.R.Addis, S.P. Bull, University of Reading

" Recent progress in Aquinas : a knowledge acquisition workbench" 2-1
J.H. Boose, J.M. Bradshaw, D.B. Shema, Boeing Computer Services

" Learning at different levels of knowledge" 3-1
Y. Kodratoff, Université de Paris-Sud,
G. Tecuci, Research Institute for Computer and Informatics, Bucharest

" Kriton : A knowledge elicitation tool for expert systems" 4-1
M. Linster, GMD

" Problems of validation in a knowledge acquisition system using multiple experts" 5-1
M. Shaw, University of Calgary

" From information analysis towards knowledge acquisition" 6-1
S. Twine, University of Queensland

" Modelling in KBS development" 7-1
G. Schreiber, J. Breuker, B. Bredeweg, B. Wielinga, University of Amsterdam

Closed session :

" Macao : A knowledge acquisition tool for expertise transfer" 8-1
N. Aussenac, J. Frontin, J.-L. Soubie, Université Paul Sabatier

" Towards a case-oriented concept for knowledge acquisition" 9-1
B. Becker, GMD

" A research framework for knowledge acquisition tools" 10-1
J.H. Boose, Boeing Computer Services

" Knowledge acquisition by observation : application to intelligent tutoring systems" 11-1
G.A. Boy, N. Nuss, Office National d'Etudes et de Recherches Aerospatiales

" Shared causal knowledge as a basis for communication between expert 12-1
and knowledge acquisition system" J. Bradshaw, University of Cambridge
R. Young, Medical Research Council

" Designing interactive expert classification system that acquires knowledge 13-1
"optimally" : an extended abstract
L.A. Cox, U S WEST Advanced Technologies

" Laps : An assistant for debriefing experts" 14-1
J.S. di Piazza, General Dynamic Corp.

" Domain independent inductive learning of relevance" 15-1
S. Dutta, University of California

" A self modifying rule-eliciter" 16-1
L. Esfahani, F.N. Teskey, Brighton Polytechnic

" Second generation knowledge acquisition systems" 17-1
B. Gaines, University of Calgary

" A new breed of knowledge acquisition systems uses redundant knowledge ?!" 18-1
M. Gams, Jozef Stefan Institute

" Classica : a knowledge acquisition system facilitating the formalization of advanced aspects in heuristic classification" 19-1
U. Gappa, Universität Kaiserslautern

" Inc^2 : A prototype-based incremental conceptual clustering system" 20-1
M. Hadzikadic, the University of North Carolina at Charlotte

" Requirements for computer aided knowledge acquisition in an ill-structured domain : a case study" 21-1
H. Frantzke, J. Herrmann, Universität Dortmund

" Wastl : An approach to knowledge acquisition in the natural language domain" 22-9
R.M. Jansen-Winkeln, Universität des Saarlandes

" Knowledge elicitation for second generation expert systems" 23-1
N.E. Johnson, C.M. Tomlinson, Brunel University

" Incremental and reversible acquisition of taxonomies" 24-1
J.-U. Kietz, Technische Universität Berlin

" Knowledge acquisition from texts" 25-1
J.U. Möller, Universität Hamburg

" Conceptual concepts of interactive knowledge acquisition tools" 26-1
M.A. Musen, Erasmus University

" Knowledge acquisition for intelligent tutoring systems" 27-1
J. Sandberg, R. Winkels, J. Breuker, University of Amsterdam

" Generation of rules with certainty and confidence factors from incomplete and incoherent learning bases" 28-1
M. Sebag, M. Schoenauer, Ecole Polytechnique

" Knowledge acquisition for a knowledge based image analysis system" 29-1
S. Schröder, H. Niemann, G. Sagerer, Universität Erlangen

" Inde : A system for knowledge refinement and machine learning" 30-1
P.P. Terptra, M.W. van Someren, University of Amsterdam

" Filling in the knowledge acquisition gap : via Kads´models of expertise to Zdest-2´s expert systems" 31-1
X. Tong, Zhejiang University,
W. Karbach, A. Voß, GMD

" Collaborative construction of an office knowledge base : Experience and suggestions" 32-1
F. von Martial, F. Victor, GMD

" Knowledge acquisition from text-based think-aloud protocols : Situational specification for a legal expert system" 33-1
T. Wetter, IBM Deutschland Gmbh,
F. Schmalhofer, Universität Freiburg

" Acquiring graphical know-how : An apprenticeship model" 34-1
D.L. Maulsby, I.H. Witten, University of Calgary

A CONCEPT LANGUAGE FOR KNOWLEDGE ELICITATION

T. R. Addis

and

S. P. Bull

Knowledge Systems Group,

Department of Computer Science,

University of Reading,

Reading, Berks.

ENGLAND

ABSTRACT

A Knowledge representation language CL is described which has evolved from Database Theory and the Philosophy of Science. The language illustrates some important features of knowledge elicitation and suggests a set of issues for further investigation. In particular, two different modes of elicitation are identified and related to types of non-deductive inference. The language makes a clear distinction between a model and the corroborating evidence that supports the model. An interpreter for the language has been constructed that provides an environment in which models may evolve in a natural manner.

1. INTRODUCTION

The representation of knowledge in a form that is eventually useful for the simulation of competence is a central issue for knowledge engineers. However, it is normally assumed that the representation scheme employed at the initial stages of knowledge elicitation will not necessarily be directly implementable. Normally there is a sequence of representation schemes which must be used to convert what is observed by the knowledge engineer into an implementation of a knowledge-based system. One of the arguments for using logic languages or production systems is their capability to be an almost direct conversion into a working simulation. It is the availability of this simulation as a prototype that

has improved the precision of the knowledge elicited by the process of feeding back to the expert the consequences of statements made about his knowledge. The simulation exposes flaws in a given expressed view of competence.

It has been proposed by Addis (Addis, 1985, Addis, 1987) that elicitation through simulation follows a similar sequence of development to that of a scientific theory (or the formation of a hypothesis). In particular, the Methodological Falsification Programme elaborated by I. Lakatos (Lakatos, 1978) describes two modes of theory formation; progressive and non-progressive. The progressive programme is where a series of theories (hypotheses) is formed. Each theory explains a set of observations and predicts further observations that may eventually be corroborated. However, there can be predicted observations that are not corroborated and the theory under investigation can prove to be inadequate for other reasons. When this occurs a new theory will be proposed that may account for more observations, suggest new potential facts and in the case of simulation may be simpler (or easier) to implement. The non-progressive programme, on the other hand, represents a different approach to dealing with predictions that are not corroborated by observation. In this programme the central theory is retained but auxiliary hypotheses are added. These auxiliary hypotheses have the effect of introducing constraints into the initial set of potential facts and restricting (or attempting to restrict) this set to those facts that can be corroborated.

Related to these two modes of theory development are two types of abduction (Peirce, 1934). The first type of abduction is where new distinctions of the world are made and it is these distinctions that form the primitives from which a hypothesis or theory can be created. These distinctions may simply be a new taxonomy of the world, or a fresh appreciation of the significance of observations that may have been overlooked or the marking of emergent relationships. This first type of abduction is linked closely with a progressive programme and is characteristic of the initial meeting between an expert and the knowledge engineer. The knowledge engineer is expected to learn a new set of distinctions from discussions, interviews and observations. The new set of distinctions is often expressed in terms of the technical language used to describe the task domain. This technical language may contain complex notions that can be expressed as a set of primitives that have to be uncovered by the

knowledge engineer. Once these primitives have been "correctly" determined, they will then provide the basis from which a simulation can be created. This simulation is the theory of the competence expressed by the expert through his statements, actions and records. It is created through the action of the second type of abduction which sets the primitives into a system that constrains the combinations of possible instances (values). The system is expected to show simplicity, symmetry and parsimony as well as confining the potential states to those that are acceptable to the expert. However, it may not be possible to design a system which displays all these characteristics. The simulation can be compared against the actions of the expert and assessed through its performance.

Once a basic simulation is available the next stage will be refinement. It is the non-progressive stage where new restrictions are imposed on the existing core simulation. Many of these restrictions will be exception rules. There may well be an extension of the basic simulation that is neither completely progressive nor non-progressive. In this intermediate case, the basic theory is embellished not by a restriction but by a new component; a component that will involve in its formation type 1 abduction.

We define knowledge elicitation as the creation of a model (a simulation) that is limited by a set of symbols. These symbols are chosen and given meaning during the first stage of the elicitation process. The model is activated through deduction and is capable of generalisation because it is defined in terms of constraints; it states what is possible which includes what is observable. However, such a model may not fit the observered data or the experts view of his own knowledge completely because it may need to be a simplification which depends upon engineering requirements. It will be a matter of negotiation between the knowledge engineer and the final user of the system as to the balance that is acceptable to each.

There are many knowledge representation schemes that may be used to specify the simulation. These knowledge representation schemes must be given in the context of some mechanism (eg an expert system shell). It is the purpose of this paper to describe a language which will:

[1] allow a knowledge engineer to establish explicitly the primitives to which meaning will already have been assigned by the user.

[2] provide a set of basic constraints that act upon these primitives to restrict their possibilities.

[3] specify a simulation that can be run directly from the statements.

[4] provide the environment which fosters the cycle of development of ideas enabling the evolution of theories (models).

[5] describe a theory that may be tested against a large database of facts.

[6] prescribe a theory which will insure that new observations are consistent with existing ones.

2. THE CONCEPT LANGUAGE

This section of the paper describes a language and its interpreter which have been specifically designed to suit the theory of knowledge elicitation described above. Related to the two types of abduction, the Concept Language (CL) comprises concepts (domain primitives) and constraints (hypothesis constructors). A set of concepts and constraints which together describe a theory are known as a concept model or the intension. Whereas, the data (or observations) that substantiate that theory are called the extension.

2.1. Concepts

A concept within the CL is a label that will serve to identify a type of object (an observation or distinction). Hence, a concept's extension is the set of known objects of a certain type, which are identified by that concept. A concept's intension describes the set of all possible objects belonging to that concept. A concept's intension comprises both the set of facets (features or attributes) that can be used to uniquely identify a member of its extension and the set of constraints that dictates whether an object is a valid member of that extension. Syntactically, a CL concept is described by the following †:

<concept> ::= <concept-label> [<facet-set>] [<constraint-set>]

<facet-set> ::= '{' {<concept>} '}'

This description becomes clearer if we consider an example. An expert at the game of chess may have

† This is a recursive definition, however, it should be pointed out that in the current implementation of the CL, concept models are restricted to three levels.

the concept of a 'chess.piece'. A 'chess.piece' is identified as being something having a name, a colour, a vertical-location and a horizontal-location. In the CL this would be stated as:

```
chess {
        piece {
                name
                colour
                vertical-location
                horizontal-location
                }
}
```

The order of facets within a concept definition is irrelevant. A concept's facets can be accessed by quoting the concept's label and a dot, followed by the facet's label. For instance, the 'name' facet in the previous example can be referenced using 'chess.piece.name'.

2.2. Constraints

As stated above, constraints constitute a part of a concept's intension. A constraint can manifest itself in one of two ways. One manifestation of a constraint is as a predicate which forms a relationship between one, two or three concepts †. A constraint in this form evaluates to true *iff* the extensions of the concepts involved are valid according to that predicate otherwise the constraint evaluates to false. The alternative manifestation of a constraint is as a set of operations involving "search", "delete" and "insert" that can be triggered to make an extension conform to the associated predicate. In the CL a constraint is expressed as follows:

<constraint> ::= ':' '(' <constraint-type> <concept-label> [<concept-label>
[<concept-label>]] ')'

A set of 16 constraints has been implemented in the current version of the CL. This set is considered to be neither a complete nor a minimal set. However, the set has been selected as being both useful and representative of the types of constraint required for concept modelling. These constraints can be classified as follows:

† We hypothesis that three is the maximum number of concepts involved in a single constraint.

Domain constraints: Although in a context-free environment it is possible to reduce any concept into a set of sub-concepts (ad infinitum), the formation of a theory (or concept model) as defined earlier requires the adoption of a set of (subjective) primitives. In this sense, a primitive is a concept that should not be capable of further reduction. The CL has built-in domain constraints for strings, atoms and numbers (integers). User-defined domains are constructed using the member(ship) constraint, which is a special form of the constraint definition given above.

Set constraints: Basic set constraints have been incorporated into the CL. These include both the binary subset and equal constraints and the ternary union, intersection, difference, and partition (exclusive union).

Mapping constraints: One-to-one, one-to-many and many-to-many mapping constraints are included in the form of bijection, reduction, combination and cross-product respectively.

Selective constraints: Finally, a special form of domain constraints, known as selective constraints have also been included. These permit the intension of one concept to be defined in terms of (or selected from) the extension of another concept. This constraint is in two forms that may be associated with the idea of analytic and synthetic knowledge. In the first case, the selection is derived through some boolean expression and in the second case, the selection depends upon what is stated to be true through interactions with the world.

3. THE CL INTERPRETER

CL is a completely declarative language. As such any concept model written in CL can be interpreted in any one of three ways:

i. Given that a set of data has been suggested as an extension of a particular concept model, we can validate this by comparing the intension with the extension and noting wherever the two are inconsistent.

ii. If a consistent (ie valid) concept model exists, it is possible to perform abduction of type 1 and type 2 from this model, thus creating a new concept model (or hypothesis); one which is either

broader in its outlook (containing new primitives) or more constrained †.

iii. Finally, the intension can be interpreted as being a description of how to simulate the related hypothesis. In order to facilitate this interpretation, instances of primitives (raw data) can be inserted (or deleted) from extensions and the subsequent consequences observed. Thus, it is possible to compare the simulated behaviour of the model with its expected behaviour. This is an essential prerequisite for the performance of abduction.

At the time of publication of this paper, only an interpreter of type three exists. Hence, of the three types of interpreter, we shall describe this one in more detail.

3.1. The Concept Simulator

The concept simulator accepts intensions as input and compiles these intensions into an internal form. Once this has occurred, data can be inserted into (or deleted from) any one of the concept extensions. In so doing, the outcome of the insertion is propagated throughout the concept model in the form of insertions and deletions to other concept extensions. Since CL constraints are directionless, insertion (deletion) can occur at any concept and propagation will continue until, either a state is reached in which the extension is consistent with the intension or a state is reached such that the extension is in direct violation with the intension and it is considered impossible for the two to be reconciled. The process of propagation is demonstrated in the following example.

Consider the game of chess. Let us expand the concept model proposed earlier to include the additional concepts of a board-position, active pieces (those still on the board), inactive pieces, white pieces and black pieces. There exist constraints between these concepts such as 'a piece is either active or inactive but not both', 'all active pieces correspond to a board-position' and 'no two pieces can exist on the same board-position'. Given that such a concept model exists and that a valid extension is known, consider the move where white captures one of black's pieces. He does this by deleting his piece from its current board-position and inserting it into the board-position occupied by the black piece. If propagation did not occur we would be left with the situation of having two pieces occupying

† In relational database terms this action corresponds to a query.

the same position. However, propagation occurs and the black piece is removed from the board. Since the black piece does not now occupy a board-position, it cannot be an active piece, hence it is deleted from the concept active-pieces. Likewise, since the black piece must still exist then it is inserted into the extension of the concept inactive-pieces.

4. EVOLVING A DESCRIPTION

The use of CL can be demonstrated if we again consider the concept of a game of chess. In the initial progressive stage of evolving a description it can be abduced (type 1) that the domain of the concept 'chess.piece.name' is constrained such that it must be one of 'king', 'queen', 'knight' etc... and similarly for colour and location. We can select from the concept of 'chess.piece' those pieces which have not been taken, this new concept is called 'active-piece'. Finally, the concept 'active-piece' is partitioned into two concepts known as 'whites-pieces' and 'blacks-pieces'. This is expressed in the CL as follows:

```
chess {
        piece {
                name : (member name '(king queen knight bishop rook pawn))
                colour : (member colour '(black white))
                vertical-location : (member vertical-location '(taken 1 2 3 4 5 6 7 8))
                horizontal-location : (member horizontal-location '(taken a b c d e f g h))
                } : (Select (<> piece.vertical-location piece.horizontal-location)
                            active-piece piece)
                  : (Partition active-piece whites-pieces blacks-pieces)
}
```

It becomes clear from this description that there are several problems, since, although all of the states of a piece may be represented, there are states that can be expressed which do not occur in practice. For example, we can have the situation where vertical-location = 'taken' and horizontal-location = 'a'. Further, there is also the possibility of stating that there is more than one piece occupying the same square.

```
chess {
        piece {
                name : (member name '(king queen knight bishop rook pawn))
                colour : (member colour '(black white))
                } : (Partition piece active-piece inactive-piece)
                  : (Partition active-piece whites-pieces blacks-pieces)
        board {
                vertical-location : (member vertical-location '(1 2 3 4 5 6 7 8))
                horizontal-location : (member horizontal-location '(a b c d e f g h))
                } : (Subset occupied board)
        positioned-piece {
                name
                colour
                vertical-location
                horizontal-location
                } : (Bijection positioned-piece active-piece occupied)
}
```

The above example has overcome the problems associated with the earlier concept model (hypothesis or theory). This has been achieved by applying type 1 abduction in order to generate new primitives (eg board, inactive-piece, occupied and positioned-piece) and by applying type 2 abduction to bring together the old and new primitives via constraints.

The consequence of this new concept model still shows too great a degree of freedom as to the kinds of states that can be expressed and a further round of development and testing will be required.

5. CONCLUSION

There have been many languages developed in attempts to capture the essential nature of knowledge and each language reflects a particular view. Unlike most of the other languages, CL has its origins in Database Theory and the Philosophy of Knowledge. The database component of the language CL no longer refers to the problems of accessing large data banks although this feature does exist within the general structure of the language. Within CL has been retained the concern for the maintenance of the data coherence that represents a world under change. Database Theory is orientated towards the construction of database interfaces that are unbiased in retrieval and that allow queries to be made without reference to the implementation; this is the particular objective of the relational view. However, CL has extended the idea of query independence to include that of update independence; it is possible to change the concept model, on-line, without the problems of engineering a new database. However, in practice there will still be an influence from the form of the original data and it will be

part of the research programme to minimise (and if possible eliminate) this effect.

The significant feature of CL which is not emphasized in other knowledge representation languages is the notion of intension and extension. This notion has the effect of making clear the distinction between the model (the theory) and the facts that are required to corroborate the model. Further, this distinction also draws out the relationship between the model and the principles from which this model has been derived (Addis, 1987). This relationship suggests that non-deductive inference (abduction and induction) can be simulated within a deductive system provided that such a system has at least three levels of intension/extension relationships. Thus the model under development is an extension of an intensional description of a learning model.

The process of knowledge elicitation is the active creation of a model. The principles by which the model is formed are expressed through the procedures followed by the knowledge engineer. These procedures have been described in the same terms as the development of a scientific theory. This general approach is not new (Gaines, 1980). However, the particular use of the Lakatosian Methodological Falsification Programme together with Peirce's views on inference applied to knowledge elicitation has brought forth new issues for consideration. These are such issues as the progressive and non-progressive cycles, the relationship between the two (or possibly three) forms of abduction and the usefulness of multi-level design of a knowledge-base system along the intension/extension dimension.

This paper has concentrated mainly on the progressive and non-progressive cycles of model development. The language CL illustrates these issues and accommodates the style of development suggested by these theoretical considerations. What is brought out by the language CL is that knowledge is expressible in terms of primitives that are fundamental to the end user and in terms of constraints that are defined within the system.

6. ACKNOWLEDGEMENTS

This work is part of the CARDS Project supported by ALVEY/SERC grant IKBS140/GR/D/7306.5 The Co-Partners in the project are Imperial Cancer Research Fund, GEC (Hirst) Research Laboratories and Reading University.

References

Addis,.

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

Addis,.

Addis, T.R., "A Framework for Knowledge Elicitation.," , p. 16, Reading University, Reading, 1987. Proceedings of the first European Workshop on Knowledge Aquisition for Knowledge-Based Systems.

Lakatos,.

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

Peirce,.

Peirce, C. S., "Scientific Method," in *Collected Papers of C. S. Peirce*, ed. A. W. Burks, Science and Philosophy, vol. vii, 1934.

Gaines,.

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*, vol. 13, pp. 81-116, 1980. Special issue on Advances in Personal Construct Technology

RECENT PROGRESS IN AQUINAS: A KNOWLEDGE ACQUISITION WORKBENCH

John H. Boose, David B. Shema, Jeffrey M. Bradshaw
Boeing Advanced Technology Center, 7L-64
Boeing Computer Services
P.O. Box 24346, Seattle, WA 98124

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. AQUINAS interviews experts directly and helps them organize, analyze, test, and refine their knowledge bases. Expertise from multiple experts or other knowledge sources can be represented and used separately or combined, giving consensus and dissenting opinions among groups of experts. Results from user consultations are derived from information propagated through hierarchies. ETS and AQUINAS have assisted in building knowledge-based systems for several years at The Boeing Company.

This paper describes recent progress on AQUINAS in the areas of knowledge base performance measurement, knowledge base maintenance, interacting trait constraints, consultation graphics, and eliciting strategic and procedural knowledge. Experiments are shown that illustrate the use of AQUINAS to automatically improve knowledge bases and even suggest new problem-solving information.

KNOWLEDGE-BASED SYSTEMS AND RAPID PROTOTYPING

Knowledge-based systems provide environments where preliminary knowledge bases from human experts may be rapidly prototyped in a matter of days or weeks. Experts, end-users, and project managers may easily examine the system's behavior and suggest changes that can be rapidly implemented. The knowledge, often stored in english-like rules, is readily available for examination, discussion, and modification by all project participants. The rapid prototyping capability of these environments is one reason why they have been successful. One of the first of these environments, EMYCIN containing the knowledge transfer tool TEIRESIAS, is still one of the best examples of such a rapid prototyping environment (Buchanan and Shortliffe, 1985; Davis and Lenat, 1982). TEIRESIAS helped knowledge engineers and experts enter rules and parameters, checked consistency, debugged consultations, and even suggested new rules. The interface and rule language could be used by the expert with a relatively small amount of training.

The major hurdle when building these systems is that of acquiring and modeling the human's expertise. A knowledge engineer - someone familiar with the knowledge-based system environment or "shell" - interviews the expert, models the knowledge, embeds it in the system, and reviews the system's behavior with the expert. The knowledge engineer changes the system based on the comments of the expert. Cycles of revision and review can take from six to twenty-four months before the system exhibits expertise when using manual interviewing methods, even in a narrow domain.

Automated knowledge acquisition tools are being developed to help cut down the revise-and-review cycle time. These tools interview experts directly and help them refine, structure, and test their knowledge. Expertise from many experts may be rapidly combined together and used. Because of its speed and flexibility, AQUINAS can be used as a more general knowledge-based decision aid, as well as a general knowledge engineering aid.

AQUINAS SYSTEM DESCRIPTION

This section discusses AQUINAS' knowledge representation and basis in Personal Construct Psychology, knowledge analysis and refinement tools, and rapid consultation prototyping capability.

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; Boose, 1988). 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 in this paper are explained more fully elsewhere (Boose, 1988; Boose and Bradshaw, 1987; Kitto and Boose, 1988; Shema and Boose, 1988). 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 that runs on a variety of hardware.

KNOWLEDGE REPRESENTED IN REPERTORY GRIDS

The expert enters and refines knowledge in the form of repertory grids. In a repertory 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. Below are excerpts from an expert building a repertory grid for a heart dysrhythmias portion of a medical aid advisor

First, the expert partitions the medical aid advisor into several sub-problems, or *cases*. Then, for the HEART.DYSRHYTHMIAS case, the expert enters a list of possible problems (termed *solutions* in AQUINAS; the goal of this part of the system is to diagnose the problem):

```
:
Please enter a list of items for expert DBS that are potential solutions for HEART.DYSRHYTHMIAS. Enter them one to a line. When you're done,
enter a RETURN.
NEW SOLUTION** NORMAL SINUS RHYTHM
NEW SOLUTION** SINUS ARRHYTHMIA
NEW SOLUTION** SINUS TACHYCARDIA
NEW SOLUTION** SUPRAVENTRICULAR TACHYCARDIA
NEW SOLUTION** SINUS BRADYCARDIA
NEW SOLUTION** ATRIAL FLUTTER
NEW SOLUTION** ATRIAL FIBRILLATION
NEW SOLUTION** NODAL RHYTHM
NEW SOLUTION** 1ST DEG HEART BLOCK
NEW SOLUTION** 2ND DEG HEART BLOCK-WENCKEBACH
NEW SOLUTION** 2ND DEG HEART BLOCK-MOBITZ II
NEW SOLUTION** 3RD DEG HEART BLOCK
NEW SOLUTION** VENTRICULAR TACHYCARDIA
NEW SOLUTION** VENTRICULAR FIBRILLATION
NEW SOLUTION** IDIOVENTRICULAR RHYTHM
NEW SOLUTION** ASYSTOLE
```

AQUINAS asks the expert for traits that help distinguish between sets of solutions.

```
Think of an important new trait that two of NORMAL.SINUS.RHYTHM, SINUS.ARRHYTHMIA, and SINUS TACHYCARDIA share, but that the
other one does not. What is that trait?
NEW TRAIT (EXTREME)** REGULAR RHYTHM

What is that trait's opposite as it applies in this case?
NEW TRAIT (OPPOSITE)** IRREGULAR RHYTHM

What is the name of a scale or concept that describes REGULAR.RHYTHM / IRREGULAR.RHYTHM?
NEW TRAIT (CONCEPT)** HEART RHYTHM

Think of an important new characteristic that two of SINUS.ARRHYTHMIA, SINUS.TACHYCARDIA, and SUPRAVENTRICULAR.TACHYCARDIA
share, but that the other one does not. What is that characteristic?
NEW TRAIT (EXTREME)** SLOW HEART RATE

What is that characteristic's opposite as it applies in this case?
NEW TRAIT (OPPOSITE)** FAST HEART RATE

What is the name of a scale or concept that describes SLOW.HEART.RATE / FAST.HEART.RATE?
NEW TRAIT (CONCEPT)** HEART RATE

Think of an important new attribute that two of SINUS.TACHYCARDIA, SUPRAVENTRICULAR.TACHYCARDIA, and SINUS.BRADYCARDIA share,
but that the other one does not. What is that attribute?
NEW TRAIT (EXTREME)** NORMAL P WAVE

What is that attribute's opposite as it applies in this case?
NEW TRAIT (OPPOSITE)** ABNORMAL P WAVE

What is the name of a scale or concept that describes NORMAL.P.WAVE / ABNORMAL.P.WAVE?
NEW TRAIT (CONCEPT)** P WAVE CHARACTERISTIC
:
```

The expert then rates each solution on each trait. By default, AQUINAS makes each trait into an ordinal trait with rating values from 1 to 5. Later, the expert can change the trait type to nominal, interval, or ratio, and change trait value ranges and intervals. AQUINAS can assist with this in certain situations, and even suggest new trait types.

```
Rate these solutions on the concept HEART.RHYTHM using a scale of 1 to 5, where 1 means more like REGULAR.RHYTHM and 5 means more like
IRREGULAR.RHYTHM. If neither one seems to apply, enter NONE If both seem to apply, enter BOTH. If you would like help estimating values,
enter USE.BARCHART. If you wish to change the type of the trait, enter CHANGE.
 REGULAR.RHYTHM(1) IRREGULAR.RHYTHM(5)
  NORMAL.SINUS.RHYTHM** 1
  SINUS.ARRHYTHMIA** 4
  SINUS.TACHYCARDIA** 1
  SUPRAVENTRICULAR.TACHYCARDIA** 1
  SINUS.BRADYCARDIA** 1
```

```
ATRIAL.FLUTTER** 2
ATRIAL.FIBRILLATION** 5
NODAL.RHYTHM** 1
1ST.DEG.HEART.BLOCK** 1
2ND.DEG.HEART.BLOCK-WENCKEBACH** 4
2ND.DEG.HEART.BLOCK-MOBITZ.II** 1
3RD.DEG.HEART.BLOCK** 1 CF .6 5 CF .4
VENTRICULAR.TACHYCARDIA** 1
VENTRICULAR.FIBRILLATION** 5
IDIOVENTRICULAR.RHYTHM** 5
ASYSTOLE** 1
  :
```

The expert rates the importance of each trait's contribution to finding a solution.

```
Please rate the relative importance of HEART RHYTHM (REGULAR.RHYTHM / IRREGULAR.RHYTHM) on a scale from 5 to 0, where 5 means more
important, and 0 means less important.
HEART.RHYTHM** 5

Please rate the relative importance of HEART.RATE (SLOW.HEART.RATE / FAST.HEART.RATE) on a scale from 5 to 0, where 5 means more
important, and 0 means less important.
HEART.RATE** 5

Please rate the relative importance of P WAVE CHARACTERISTIC (NORMAL.P.WAVE / ABNORMAL.P.WAVE) on a scale from 5 to 0, where 5
means more important, and 0 means less important.
P.WAVE.CHARACTERISTIC** 4
  :
```

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 solutions and traits are analyzed to help the expert refine useful distinctions and eliminate those that are inconsequential or redundant. For example, NORMAL.SINUS.RHYTHM and 1ST.DEG.HEART.BLOCK are found to be highly matched, and the expert enters a new trait, PR.INTERVAL, to help further distinguish between them. After the new row is added, AQUINAS asks the expert to rate all the solutions on the new trait.

```
The two solutions NORMAL.SINUS.RHYTHM and 1ST.DEG.HEART.BLOCK are matched at the 95% level. Can you think of some new trait that
would distinguish between them?
AQU** YES

What is the name of that trait?
NEW TRAIT (CONCEPT)** PR.INTERVAL
  :
```

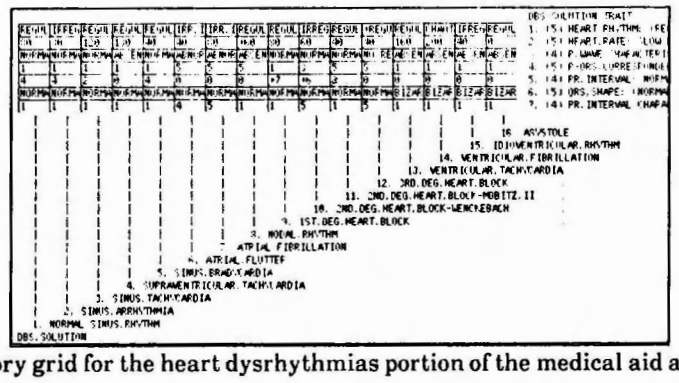


Figure 1. Repertory grid for the heart dysrhythmias portion of the medical aid advisor.

Inductive implications between trait values can show the expert higher levels of abstraction implied by a repertory grid (Figure 2). If the expert disagrees with an implication, AQUINAS helps the expert refine the grid. This is one way in which interactive learning takes place in AQUINAS.

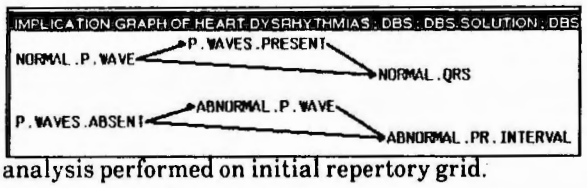


Figure 2. Implication analysis performed on initial repertory grid.

Trait scale types may be changed from the original default (ordinal scale with ranges from one to five) to other types (nominal, interval, ratio scales with user-defined ranges) based on precision needed, cost of elicitation, and representation convenience. AQUINAS helps the expert change trait scale types in several ways. Here, AQUINAS notices that one trait has only extreme rating values (all ratings are either 1 or 5), and suggests that the trait scale type be changed from ordinal to nominal. The expert agrees, and AQUINAS automatically maps the numerical values to symbolic nominal values.

P.WAVE CHARACTERISTIC has only extreme values. Should its trait type be changed from ORDINAL to NOMINAL?
AQU** **YES**
Use the default method for mapping old values to new values of P.WAVE.CHARACTERISTIC?
AQU** **YES**

Mapping for NORMAL.SINUS.RHYTHM from 1 to: NORMAL P WAVE 1.00 DEFAULT MAPPING
Mapping for SINUS.ARRHYTHMIA from 1 to: NORMAL.P WAVE 1 00 DEFAULT MAPPING
:

Nominal, ordinal, and interval trait scales were illustrated in the grid in Figure 1.

HIERARCHICAL REPERTORY GRIDS

In a major extension to Kelly's methods, experts use AQUINAS to arrange repertory 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 3 and 4).

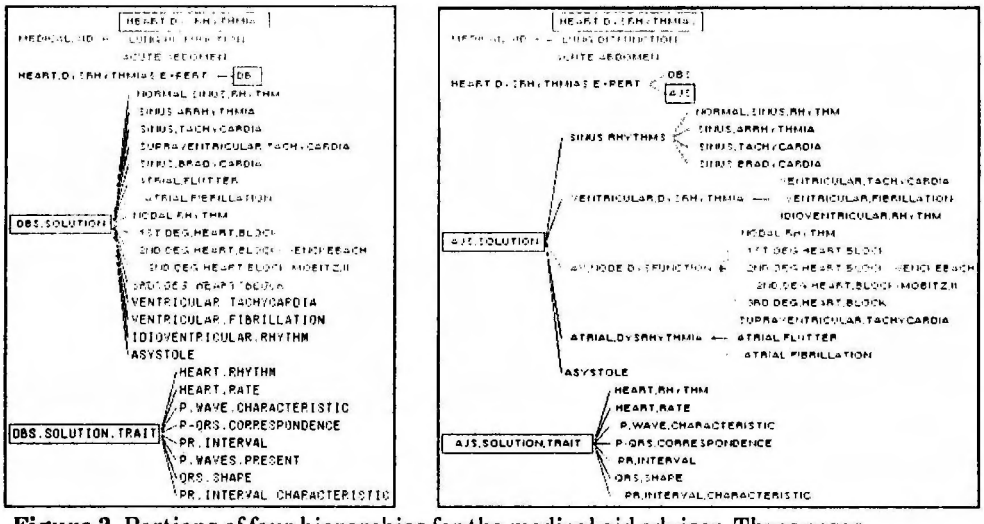
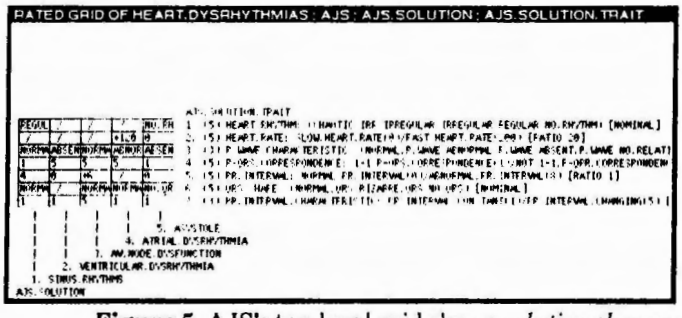


Figure 3. Portions of four hierarchies for the medical aid advisor. Three cases are shown, as well as expert DBS's solution and trait hierarchies. So far, DBS is the only one who has entered information about heart dysrhythmias.

Figure 4. Another expert, AJS, has entered knowledge for the same case. Tools in AQUINAS were used to help AJS structure solutions in a class hierarchy. The hierarchies are related from top to bottom: given the HEART.DYSRHYTHMIAS case, experts DBS and AJS exist (other experts could exist for other cases). Given expert AJS, we see his solution hierarchy (the top of the hierarchy is the class AJS.SOLUTION). Given that we are looking at the top-level grid in the hierarchy (the children of AJS.SOLUTION), the immediate children of AJS.SOLUTION.TRAIT apply. Different traits can apply to different class levels in the solution hierarchy; different solutions could come from different experts, and so on.

Nodes in the four hierarchies combine to form repertory 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 (Figure 4). Rating values within the grid are judgments about the solutions with respect to each trait (Figure 5).



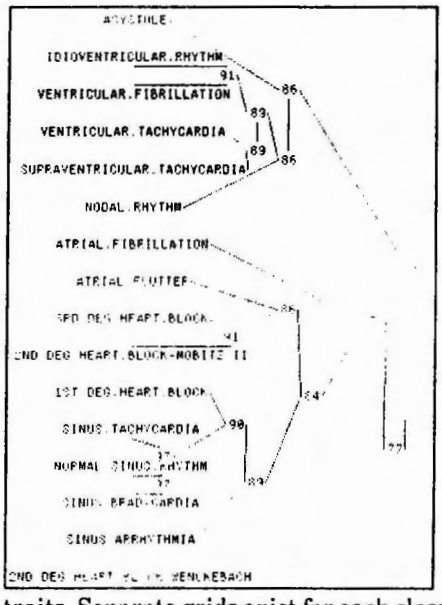


Figure 5. AJS's top-level grid shows *solution classes* and traits. Separate grids exist for each class. During consultations, AQUINAS will search the hierarchies to ask questions and infer ratings.

Figure 6. A solution cluster analysis is produced from AJS's initial repertory grid. AJS is asked to form hierarchies by labeling junctions on the cluster. The clusters SINUS.RHYTHMS, VENTRICULAR.DYSRHYTHMIA, AV.NODE.DYSFUNCTION, and ATRIAL.DYSRHYTHMIA, shown in Figure 4, are formed.

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 6), *implication pattern analysis* (certain patterns may suggest levels of hierarchical subsumption), and *trait value examination* (certain combinations of trait values may suggest that hierarchical expansion is appropriate).

DECISIONS AND PERFORMANCE: CONSULTATIONS IN AQUINAS

The knowledge embedded in hierarchical repertory 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 maximum entropy approach is applied by AQUINAS's reasoning mechanism (Boose and Bradshaw, 1987). ETS used a reasoning approach based on rules and certainty factors (Boose, 1986a).

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).

Following are portions of a transcript from an AQUINAS consultation using expertise from both DBS and AJS. The user specifies trait preferences, and AQUINAS combines the preferences together producing consensus and dissenting opinions:

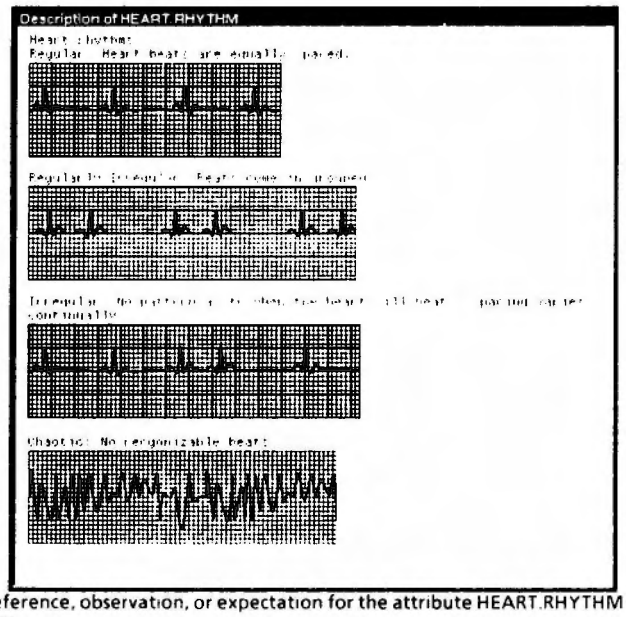
```
  :
Enter a brief description of this consultation
AQU** 56 YO MALE FOUND DOWN, UNKNOWN DOWNTIME, NO CPR @ ARRIVAL

What solutions would you like to consider for consultation MEDICAL-AID--ALLEN ? Enter them one to a line If you wish every solution to be
considered, enter ALL. You may specify solution weights by appending selections with a number between 0 and 1 (e.g., .25). When done, enter
RETURN.
AQU** ALL

The following experts know about the solutions : DBS AJS
Would you like to weight these experts?
AQU** NO

What kind of ordering should be used for questions (COST-BENEFIT SEQUENTIAL USER-SPECIFIED)?
AQU** SEQUENTIAL
  :
```

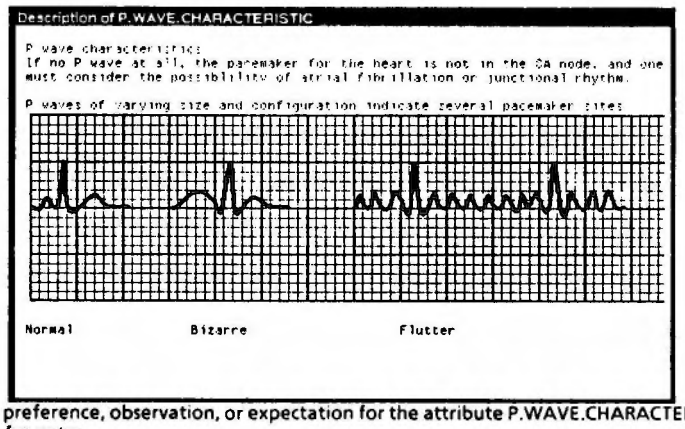
AQUINAS can display graphic information when asking questions. Here, the user answers a consultation question by comparing a scope rading with categories of heart rhythm.



Please indicate your preference, observation, or expectation for the attribute HEART.RHYTHM (AJS) You may also USE.BARCHART for entry.
(CHAOTIC IRR.IRREGULAR IRREGULAR REGULAR NO.RHYTHM) WT 5
AQU** **REGULAR**

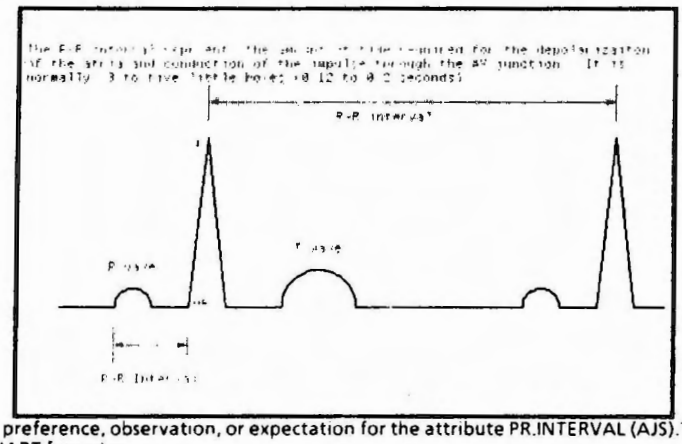
Please indicate your preference, observation, or expectation for the attribute HEART.RATE (AJS).The interval is 20 You may also USE.BARCHART for entry.
SLOW.HEART.RATE(0) / FAST.HEART.RATE(200) WT 5
AQU** **60**

Again, the user enters information by comparing a scope reading with graphic information displayed by AQUINAS. The expert or knowledge engineer enters this information by preparing a graphic file that is associated with a trait. Graphics can be created with a variety of paint programs and bitmap editors, or scanned and digitized. Six graphic displays are associated with this knowledge base; they took approximately two hours to enter.



Please indicate your preference, observation, or expectation for the attribute P.WAVE.CHARACTERISTIC (AJS). You may also USE.BARCHART for entry.
(NORMAL.P.WAVE ABNORMAL.P.WAVE ABSENT.P.WAVE NO.RELATION) WT 3
AQU** **NORMAL.P.WAVE**

Please indicate your preference, observation, or expectation for the attribute P-QRS.CORRESPONDENCE (AJS). You may also USE.BARCHART for entry.
1-1.P-QRS.CORRESPONDENCE(1) / NOT.1-1.P-QPR.CORRESPONDENCE(5) WT 5
AQU** **1**



Please indicate your preference, observation, or expectation for the attribute PR.INTERVAL (AJS).The interval is 1 You may also USE.BARCHART for entry.
NORMAL.PR.INTERVAL(0) / ABNORMAL.PR.INTERVAL(8) WT 5
AQU** **4**
:—

After the observations are entered, AQUINAS displays the results of the consultation. Weighted contributions from each expert are shown. When only one expert contributes to a solution, the weight is shown at 1.00 (for example, SINUS.RHYTHMS). When more than one expert contributes to a solution each expert's contribution weight is shown. In SINUS.BRADYCARDIA for example DBS and AJS each contribute equally (.5) so a weighted average of their contributions is computed as the total weight for the solution. If the user had weighted the experts differently, this would be taken into account when computing the average.

Test results for consultation
MEDICAL-AID--ALLEN

1 : SINUS.RHYTHMS (0.98)	(AJS 1.00 0.98)
2 : NORMAL.SINUS.RHYTHM (0.91)	(DBS 0.50 0.83)(AJS 0.50 0.99)
3 : SINUS.BRADYCARDIA (0.90)	(DBS 0.50 0.83)(AJS 0.50 0.98)
4 : SINUS.TACHYCARDIA (0.87)	(DBS 0.50 0.77)(AJS 0.50 0.96)
5 : SINUS.ARRHYTHMIA (0.83)	(DBS 0.50 0.83)(AJS 0.50 0.83)
6 : 1ST.DEG.HEART.BLOCK (0.73)	(DBS 0.50 0.78)(AJS 0.50 0.68)
7 : 2ND.DEG.HEART.BLOCK-MOBITZ.II (0.64)	(DBS 0.50 0.61)(AJS 0.50 0.68)
8 : AV.NODE.DYSFUNCTION (0.62)	(AJS 1.00 0.62)
9 : SUPRAVENTRICULAR.TACHYCARDIA (0.59)	(DBS 0.50 0.56)(AJS 0.50 0.62)
10 : 2ND.DEG.HEART.BLOCK-WENCKEBACH (0.57)	(DBS 0.50 0.61)(AJS 0.50 0.52)
11 : NODAL.RHYTHM (0.56)	(DBS 0.50 0.48)(AJS 0.50 0.64)
12 : ATRIAL.DYSRHYTHMIA (0.56)	(AJS 1.00 0.56)
13 : ATRIAL.FLUTTER (0.50)	(DBS 0.50 0.39)(AJS 0.50 0.60)
14 : 3RD.DEG.HEART.BLOCK (0.48)	(DBS 0.50 0.36)(AJS 0.50 0.59)
15 : VENTRICULAR.FIBRILLATION (0.46)	(DBS 0.50 0.55)(AJS 0.50 0.38)
16 : ASYSTOLE (0.45)	(DBS 0.50 0.45)(AJS 0.50 0.45)
17 : IDIOVENTRICULAR.RHYTHM (0.43)	(DBS 0.50 0.48)(AJS 0.50 0.38)
18 : ATRIAL.FIBRILLATION (0.40)	(DBS 0.50 0.36)(AJS 0.50 0.44)
19 : VENTRICULAR.TACHYCARDIA (0.40)	(DBS 0.50 0.42)(AJS 0.50 0.38)
20 : VENTRICULAR.DYSRHYTHMIA (0.38)	(AJS 1.00 0.38)

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 this case, both experts exhibited moderately strong agreement with the consensus.

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 the results below, the first several solutions are in substantially the same order. The user gains more confidence that these may be valid recommendations. If there are wide differences between experts, the user can employ other tools in AQUINAS to compare the appropriate repertory grids to see where the experts' opinions differed.

Dissenting opinion for consultation MEDICAL-AID--ALLEN

Correlation scores for all experts:
DBS 0.86
AJS 0.84

AJS has the most dissenting opinion (0.84 correlation score) ,

AJS	/ Consensus
1 : NORMAL.SINUS.RHYTHM 0.99	/ SINUS.RHYTHMS 0.98
2 : SINUS.RHYTHMS 0.98	/ NORMAL.SINUS.RHYTHM 0 91
3 : SINUS.BRADYCARDIA 0.98	/ SINUS.BRADYCARDIA 0.90
4 : SINUS.TACHYCARDIA 0.96	/ SINUS.TACHYCARDIA 0.87
5 : SINUS.ARRHYTHMIA 0.83	/ SINUS.ARRHYTHMIA 0.83
6 : 1ST.DEG.HEART.BLOCK 0.68	/ 1ST.DEG.HEART.BLOCK 0.73
7 : 2ND.DEG.HEART BLOCK-MOBITZ.II 0.68	/ 2ND.DEG HEART BLOCK-MOBITZ.II 0.64
8 : NODAL.RHYTHM 0.64	/ AV.NODE.DYSFUNCTION 0.62
9 : SUPRAVENTRICULAR.TACHYCARDIA 0.62	/ SUPRAVENTRICULAR.TACHYCARDIA 0.59
10 : AV.NODE.DYSFUNCTION 0.62	/ 2ND.DEG HEART BLOCK-WENCKEBACH 0.57
11 : ATRIAL.FLUTTER 0.60	/ NODAL.RHYTHM 0 56
12 : 3RD.DEG.HEART.BLOCK 0.59	/ ATRIAL DYSRHYTHMIA 0 56
13 : ATRIAL.DYSRHYTHMIA 0.56	/ ATRIAL FLUTTER 0.50
14 : 2ND.DEG.HEART.BLOCK-WENCKEBACH 0 52	/ 3RD.DEG HEART BLOCK 0 48
15 : ASYSTOLE 0.45	/ VENTRICULAR.FIBRILLATION 0 46
16 : ATRIAL.FIBRILLATION 0.44	/ ASYSTOLE 0.45
17 : VENTRICULAR.DYSRHYTHMIA 0.38	/ IDIOVENTRICULAR.RHYTHM 0.43
18 : VENTRICULAR.TACHYCARDIA 0.38	/ ATRIAL.FIBRILLATION 0.40
19 : IDIOVENTRICULAR.RHYTHM 0.38	/ VENTRICULAR.TACHYCARDIA 0.40
20 : VENTRICULAR.FIBRILLATION 0.38	/ VENTRICULAR.DYSRHYTHMIA 0.38

CONSTRAINTS

In AQUINAS, *constraints* are interactions between traits specified by the expert or by the end user. Use of constraints allows the expert to show relationships between traits and allows users to express absolute solution expectations as well as preferences.

Experts can enter relationships that constrain traits given the values of other traits. These types of constraints can limit the solution set, prune questions, and affect question ordering during consultations. For example, the following response triggered an expert-supplied constraint. If the heart rate is 40 or below, certain rhythms won't occur. If this happens during a consultation, these rhythms are pruned from the solution candidates (their scores are set to zero and not changed). The expert also entered the constraint that the rhythm asystole is eliminated for all heart rates greater than zero. In another example, the expert entered a constraints that prunes traits during consultations. In effect, this stops certain questions from being asked. For instance, if the P.WAVE. CHARACTERISTIC is ABSENT.P.WAVE, then questions about PR.INTERVAL and PR.INTERVAL.CHARACTERISTIC are not asked. These types of constraints entered by the expert can also affect question ordering.

The user can also enter constraints during a consultation. Here, for example, the user constrains the solution set to those with heart rates greater than 120 by using the keyword ABSOLUTE. Any solutions associated with heart rates less than 120 will be pruned:

```
Please indicate your preference, observation, or expectation for the attribute HEART.RATE (DBS).The interval is 20  You
may also USE.BARCHART for entry.
 SLOW.HEART.RATE(0) / FAST.HEART.RATE(200) WT 5
AQU** 120 > ABSOLUTE
```

The user can also enter contraints for single values:

```
Please indicate your preference, observation, or expectation for the attribute HEART.RHYTHM (DBS)  You may also
USE.BARCHART for entry.
 (REGULAR IRREGULAR IRR.IRREGULAR CHAOTIC) WT 5
AQU** REGULAR ABSOLUTE
```

The constraints established by the user, HEART.RATE greater than 120 and a HEART.RHYTHM of regular, eliminate most of the solutions (all other solutions received scores of zero):

```
Test results for consultation  MEDICAL-AID--EXAMPLE2

1 : SINUS.TACHYCARDIA ( .99 )
2 : SUPRAVENTRICULAR.TACHYCARDIA ( 0.83 )
3 : VENTRICULAR.TACHYCARDIA ( 0.69 )
```

The addition of constraints to repertory grids has allowed AQUINAS to represent, in general, any type of information that could be represented in rules.

CONSULTATION REVIEW MECHANISM: KNOWLEDGE BASE DEBUGGING AND MAINTENANCE; VALIDATION AND VERIFICATION

A general problem when modifying knowledge bases is that changes may degrade system performance. This is especially a problem when the knowledge base is large; it may be unclear how changing one item in a knowledge base containing thousands of items will affect overall system performance. A set of consultation review tools in AQUINAS helps alleviate this problem.

Tools for analyzing consultations to improve performance are an important part of the workbench (Shema and Boose, 1988). These tools may modify the knowledge base after checking that existing test cases still give the intended

results. Synthesis of traits and test cases help with the validation of the knowledge base. Experiments are underway to measure the utility of these tools.

Case Storage and Playback

Information entered by the expert during a consultation may be stored for later playback. The recording and playback facilities test new changes to the knowledge base using old consultations. The description of the case, the values of the traits, and the expert's expectations of the ranking are recorded with each stored case.

Cases can be run in batch mode, where changes made to the knowledge base to improve the performance of one test case may be automatically tested on all the previous cases. The expert can see whether or not new changes result in overall improvement.

Knowledge Base Changes

Changes that lead to knowledge base performance improvement (or degradation) take several forms:

Changing ratings in a grid. Judgments about the relationships between solutions and traits may be revised.

Changing a trait's weight. A trait's influence may be increased or decreased by changing its weight. Traits can be temporarily ignored by assigning them a weight of zero.

Adding or deleting a trait. Traits that affect the outcome may be added, deleted, or moved to different levels in the hierarchies. New traits are rated against existing solutions and assigned weights.

Adding or deleting a solution. The potential set of solutions that affect the outcome may be expanded or restricted. New solutions are rated against existing traits.

Changing the expectations of outcomes. The expert may revise the expectation of the system's behavior.

Re-organizing existing knowledge. Knowledge can be reorganized to change its affect on performance. For instance, knowledge can be regrouped under different cases in the knowledge bases hierarchies so that it is applied under different circumstances. Experts may also change their ideas about the range of applicability of portions of the knowledge base.

Review mechanisms fall into several categories - *manual, directed, semi-automated,* and *automated.*

Manual Methods - Inspect Results and Revise

A variety of methods exist to manually inspect consultation results. The expert may review the results and change the knowledge base directly to try and improve the system's performance.

The rank ordering of the solutions may be displayed in list form (as above), tables, bar charts, or pie charts. Tracing of an ongoing consultation can be done graphically using bar charts to display the current stage of the ranking, or tracing information can be displayed as text. The amount of tracing can be controlled by the expert.

Dynamic bar charts show intermediate consultation results. The effects of each trait can be seen on the solution set. After watching the effect of a particular trait, the expert may manually adjust the repertory grid information to decrease or increase the effect of the trait on the solution set.

Final consultation results may be displayed as an ordered bar chart. This graphic representation helps the expert see the *intervals* between solutions, while textual lists help the expert see the *overall ordering*. Consultation results can also be displayed in normalized form in a pie chart. It is often easier to see the *relative weight* of each solution in a pie chart (as opposed to the *absolute weights* emphasized in a bar chart).

Each individual rating's contribution to the solution may be emphasized by highlighting those grid elements which come closest to preference ratings given during a consultation. The expert can quickly see the information that contributed to the particular test case results.

All of these lists, graphs, charts, and grids may be saved from consultation to consultation. The expert can compare previous results to current results and see if changes led to improvements. These displays can also be recreated from saved consultations.

Directed Methods - Elicit Results Expectations, Score Results, Revise

The correlation between the expert's expectation of the solution ranking vs. the outcome of a consultation may be calculated. This correlation is used as a measure of the quality of the knowledge base performance for a case. Solutions that are out of place can be used to elicit new information. The expert may partially or completely specify the expected outcome of a consultation.

Scoring results. Results are scored against expectations by computing a rank correlation measure and checking to see whether or not the result is statistically significant. Correlation scores show the expert how well the knowledge base performed, given the expectations. Correlation scores recorded over time for many test cases can show whether or not the knowledge base is improving as revisions are made.

New Triads Based on Expectations. AQUINAS calculates the solution that is most out of place and asks the expert for a new trait based on where the solution *should* have appeared in the list. When this new trait is included in a subsequent consultation the out-of-place solution should be closer to the expected place in the list (Boose, 1986).

Semi-Automated Methods - Suggesting Changes

Solution results adjustment. Implementation of these methods is in progress. If the expert disagrees with the rank ordering of the solutions, he will be able to modify the bar chart displaying the results of the consultation. His new ordering will be used to find potential changes to ratings or trait weights that would lead to the new ranking. Changes to *repertory grid values* may be suggested that would help push target solutions up or down on the results list. Changes to *trait weights* may be suggested that would help push groups of solutions up or down on the results list. After information

is modified the consultation will be replayed and the new and old results will be compared. This method of interaction is similar to a spread-sheet: the expert modifies the answer graphically and the data is used to calculate changes.

Automated Methods - Rating and Trait Weight Revision; Synthesizing New Traits

Preliminary results are presented for methods that examine the knowledge in a repertory grid and automatically generate changes that lead to performance improvements.

Automatic Rating and Weight Adjustment. Values of ratings and trait weights may be varied automatically to produce a new repertory grid. Batch test cases are run, and the results of the changes are measured against the expert's expectations. A new change is selected, and the test cases are re-run and scored.

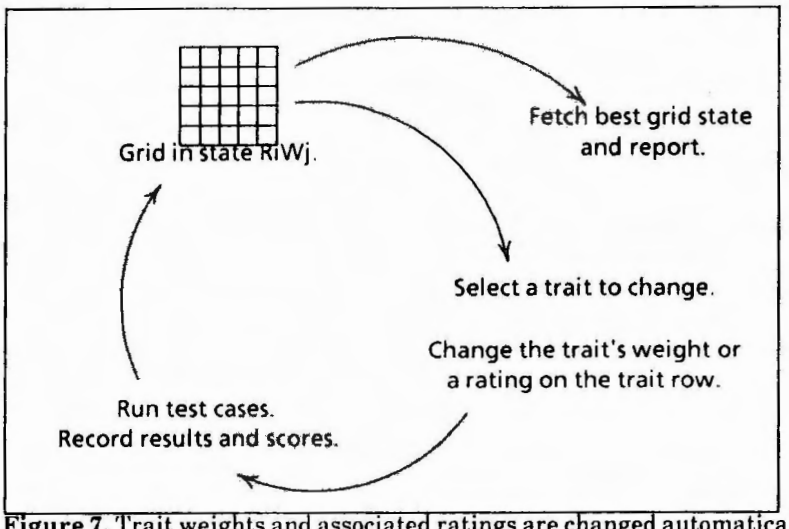


Figure 7. Trait weights and associated ratings are changed automatically, batch test cases are run, and the results are scored against the expert's expectations for those cases and stored. Heuristics are used to select the next trait-weight-rating change. When finished, the best combination of trait weights and ratings is reported to the expert. "Best" is defined as the highest average correlation score over the test case set for that grid state. The expert can examine the proposed grid and direct AQUINAS to implement all or a portion of the suggested changes.

Figure 7 shows the grid modification, test, score, and report cycle. The expert starts by specifying a set of test cases and expected outcomes (a rank-ordered solution set for each test case).

Exhaustively checking every possible combination of ratings and weights is not feasible for even medium-sized grids. For example, it took over ninety hours to test a small seven-by-six grid, sampling ratings (1, 3, and 5 on an ordinal scale from 1 to 5), and using six test cases. So, heuristics are used to select a new change and decide when to stop work. Heuristics are based on:

- the original trait weight (traits with higher weights tend to be tried sooner),
- the original rating distribution spread in the row,
- the distance from the original grid state to the proposed new grid state (closer states are tried first since it is assumed that the grid from the expert is reasonably close to a final state), and
- previous results (if moving a rating or weight in a certain direction produces better results, try pushing it farther in that direction).

We are presently trying a combination these heuristics and of hill-climbing. Starting with the expert's grid, one change to a rating or weight is made. The change is selected based on the distance of the proposed change from the original grid; "closer" changes are tried before more distant changes. If the change results in overall performance improvement, a change is made from this new state. If the change does not result in an overall improvement, a different change is tried from the old state. If no changes result in performance improvement, a "peak" has been reached. Random grid changes are tried periodically to keep from only finding local minima; if the random grid produces an overall performance improvement, hill climbing proceeds from there. This process runs until a user-specified time limit is up.

When the process is over, several results are reported:

The grid that exhibited the best performance. The expert can examine the new grid to see whether or not the changes make sense. If the expert agrees, all or some of the changes are automatically implemented in the existing grid.

The "distance" between the best grid and the original grid. Trait ratings and weights in the new grid may be slightly different or significantly different from those in the original grid. When there are significant differences, it may suggest that a new trait should be used that exhibits the new ratings and weights (see the discussion of trait synthesis, below).

A sensitivity ranking for the traits. The overall effect each trait has on all the tested combinations is reported. Traits are ranked according to the greatest performance variance and reported. This information can help focus the expert's attention on traits that most affect performance.

Local minima. Using the hill-climbing approach described above, grids producing local minima are also reported. These grids may contain optimal interactions between a subset of traits.

A "completeness" score for the test case results. A score is given that shows the overall completeness of possible test case results rankings for the best (or any other) grid (for instance, there would be twenty-five possible

outcomes for five possible solutions). This helps the expert see how well the test cases cover the set of solutions. This information can be used to generate new potential test cases (see below).

This technique has produced promising results. Minor sets of changes have yielded significant problem-solving improvement over small sets of test cases.

Analytic and Synthetic Changes. Using AQUINAS, the best case of knowledge base modification (from a maintenance point of view) is when adjustments made solely to trait values and weights result in improvements of (or at lease, do not degrade) all existing cases. These are referred to as *analytic* changes, since no new traits or solutions are added, and a change can be made by analyzing relationships between existing traits and solutions. Usually this is not the case; a given change may improve some case performance and degrade other case performance. When this happens, knowledge must be added and/or deleted. Adding new knowledge is referred to as a *synthetic* change.

Synthesizing New Traits. Examination of the consultation results makes it possible to hypothesize new traits that would continue to keep the knowledge base correct for existing cases. A new trait is formed internally for a set of solutions; its values are systematically varied and combined with existing traits and tested against existing consultations. If the inclusion of this trait improves the overall scores of existing consultations, the expert is asked to try and name it, and it is included in the knowledge base. Such traits can also be used to start a *new* grid (Shema and Boose, 1988).

Implementation of the following two methods is in progress.

Consultation synthesis. Examining the contents of previous consultations may suggest new test cases that are significantly different from those already present (that is, their rank orders are significantly different from any existing test cases). The expert could evaluate these synthesized test cases and try to see if they fit any real-world situations. The knowledge base validation effort could be assisted by the expanded coverage of these test cases.

Test Case Analysis. Comparisons of consultation results might highlight discrepancies in the expected results that the expert has given. Test cases may not be appropriate for the section of the knowledge base under consideration, or the expert may have made an error in his ranking. Validation of the test cases often is as important as the validation of the knowledge base.

STRATEGIC AND PROCEDURAL KNOWLEDGE

We have performed experiments with AQUINAS to elicit high level strategic knowledge. We took cases and listed them on the top of the grid, and then asked the expert, "What strategy might you use to solve two of CASE1, CASE2, and CASE3, but not the third?" Strategies were elicited as traits, and then the expert rated the applicability of each strategy to each problem case. This method tended to highlight the priorities of various procedures in solving the problem (Figures 9 and 10).

What strategy might you use to solve two of AUTO.ACCIDENT, MEDICAL.CALL, and MOUNTAIN.RESCUE, but not the third?
NEW STRATEGY** **FIND OUT LOCATION**

What is that strategy's opposite as it applies in this case?
NEW STRATEGY** **LOCATION NOT ESSENTIAL**

What is the name of a scale or concept that describes FIND.OUT.LOCATION / LOCATION.NOT.ESSENTIAL?
NEW STRATEGY (CONCEPT)** **LOCATION**

What strategy might you use to solve two of MEDICAL.CALL, MOUNTAIN.RESCUE, and MISSING.CHILD, but not the third?
NEW STRATEGY** **FIND OUT SEVERITY**

What is that strategy's opposite as it applies in this case?
NEW STRATEGY** **SEVERITY NOT ESSENTIAL**

What is the name of a scale or concept that describes FIND.OUT.SEVERITY / SEVERITY.NOT.ESSENTIAL?
NEW STRATEGY (CONCEPT)** **SEVERITY**

:

Figure 9. Triad comparison of cases is used to elicit strategies and procedures for a hot-line dispatch portion of the medical aid advisor.

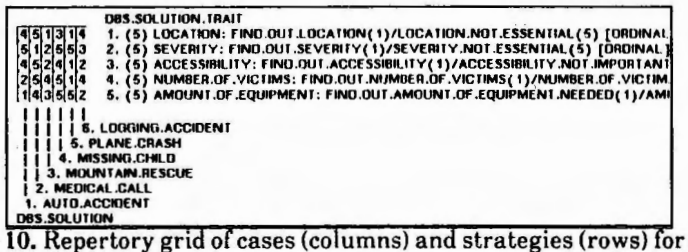


Figure 10. Repertory grid of cases (columns) and strategies (rows) for a hot-line dispatch portion of the medical aid advisor. For this problem, strategies are *procedural priorities* for determining different types of information.

When this information is inverted (rows and columns are exchanged so that strategies become "solutions" and cases become "traits") the type of case can be used to select the appropriate strategy during consultations.

AUTOMATICALLY GENERATING PRODUCTION RULES

AQUINAS can transform repertory 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). 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.

Experts find it easier to analyze and maintain knowledge in the more compact form of repertory grids rather than in the form of production rules. In ETS, use of production rules and an internal reasoning engine were the sole methods of reasoning. This method has been replaced in AQUINAS with a maximum entropy-based approach. Rule-based reasoning in AQUINAS is still used to pre-test knowledge bases destined for other expert system building tools. Certainty factor generation has been fine-tuned so that results using maximum entropy and certainty factors are very close for most applications.

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, 1987b, 1988). 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

AQUINAS APPLICATION AREAS

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, AQUINAS can often be used on the analytic components of larger synthesis problems.

DIMENSIONS OF AQUINAS USE

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 repertory grids and because of its ability to rapidly gather knowledge and build a prototype consultation system.

Dimensions of use of AQUINAS as a knowledge-based decision making aid are listed below.

Source and user: The source of the knowledge may be the same or different than the user of the knowledge.

Source quantity: One person or more than one person may contribute to the knowledge base.

User quantity: One person or more than one person may use the knowledge base.

Knowledge base longevity: Knowledge bases may be used once and "thrown away" after some problem insight has been achieved or they may be used repeatedly, at different rates (continuously, hourly, daily, weekly, monthly, and so on).

Knowledge base update frequency: Knowledge may be updated with varying rates of frequency. For example, a "frozen" knowledge base may be used for a long time before it is updated or a "living" knowledge base may be updated continuously (a large, commercial, time-share knowledge base with few updates; a marketing trends knowledge base with periodic updates; a stock market knowledge base with continuous updates).

Stand-alone: The tool can be used for stand-alone decision use or used with other software systems (expert system shell; spreadsheet; database management system).

Commercial knowledge base use: The knowledge base can be generated for commercial, external or internal, or personal use.

Consulting Use: The decision maker can use the tool directly or a consultant can use the tool in support of the decision maker (decision consulting; education and training).

Empty tool or primed tool: The tool can be used as an empty general decision aid or primed with knowledge for specific applications. Knowledge templates or full or almost-full knowledge bases may be delivered to a customer. Some examples are knowledge for specific domains at varying levels of abstraction (general diagnosis, engine diagnosis, jet engine diagnosis), proprietary or commercially valuable knowledge, surveys (fill in the ratings and return the grid to the surveyors), or use of domain-tailored dialogs (such as the original use of repertory grids in psychotherapy where patients filled in specific roles based on supplied descriptions).

Process and product: These tools can be used in the process of making a knowledge base as well as actually delivering knowledge (feasibility analysis, education and training of experts in the knowledge engineering process).

ETS (based on single grids) and AQUINAS (handling multiple hierarchical grids) have been used to generate hundreds of small and medium sized knowledge-based systems in the following categories (Boose, 1988). The categories are derived using different combinations of values for the above dimensions.

1. **Decision aid for for one-shot decisions**
   A grid tool is used to help people gain insight while making decisions: What employee should I assign to this project? What stocks should I invest in? What car should I buy? How can I better represent my products to my customers?
2. **Stand-alone knowledge-based system development and delivery tool**
   A grid tool is used to help develop, deliver, and maintain a knowledge-based system in a cost-efficient manner.
3. **Group decision aid**
   A grid tool is used as an aid to facilitate rapid, documented group decisions: On what software and hardware environment should this project be implemented? What research should our company pursue? How should capital assets be assigned?
4. **Large-scale on-line knowledge bases for a community of users**
   A grid tool can be used to develop and maintain large dial-in knowledge-and-data bases for a large number of users (stock market advising, insurance claims information).
5. **Feasibility analysis and project exploration tool**
   A grid tool is used to help assess project feasibility: What project idea is technically feasible? What project ideas are possible in this general area? Given a project idea, what kinds of knowledge will be involved? Does it make sense to invest resources in this project?.
6. **Expert system building tool (shell) front-end**
   A grid tool can be used as a front-end to develop knowledge for another expert system shell (S.1, M.1, OPS5, etc.).
7. **Teaching aid**
   A grid tool can be used to help teach others (especially experts and beginning knowledge engineers) quickly and painlessly about the concepts involved in knowledge-based systems and knowledge engineering: What are production rules? What is a consultation system? How can an expert's knowledge be structured and modified? How can systems be tested and validated?
8. **Situation Insight**
   A grid tool can be used to discover important or controlling factors in a situation; these factors can help the user understand, control, or change the situation (job satisfaction and motivation self-analysis, psychotherapy, counseling, relations with colleagues).

Large knowledge bases (containing thousands of judgments) are typically generated as the result of group decision applications. A twenty-by-twenty grid contains 400 ratings; when such grids a collected from multiple experts the resulting knowledge base contains many thousands of ratings. For example, a past AI tool advisor system included over four thousand ratings from four experts. Larger systems from single users tend to have more hierarchical structure and fewer ratings.

USING AQUINAS

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).

Alternative approaches may be tested with little expenditure of resources.

The first expert took about two hours to enter, analyze, and refine the knowledge for the heart dysrhythmias case shown here. Working from the first expert's solutions and traits (but not ratings or weights), the second expert took about an hour-and-a-half to enter, analyze, and refine the knowledge for this case. A knowledge engineer took less than two hours to draw and enter the graphic displays.

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

In the future we plan to extend AQUINAS in several areas:

Capturing Explanations. Templates will be used to capture comments from experts when making judgments (ratings, weights, traits from triads). Comments will be examined to help maintain the knowledge base and will be played back during consultations to provide explanations for the end-user. Applicable work on classifying explanations has been reported by Schank (1986) and Kass and Leake (1987).

Further Work in Acquiring and Analyzing Knowledge from Multiple Experts. In addition to AQUINAS' current capabilities to analyze and use knowledge from multiple experts, we plan to implement the following features:

- Elicit knowledge in parallel (on-line) from several experts (Chang, 1986) in the same or different fields.
- Analyze subsumption and overlap of experts' knowledge in multiple fields (similar to SocioGrids in KSS0, Shaw and Gaines, 1987).
- Support on-line structured negotiation among experts to solve problems or make recommendations.
- Discover the most important aspects of individual knowledge when compared to group knowledge (special cases).
- Elicit and merge knowledge dynamically for use in on-line applications.

Knowledge Acquisition for Synthesis Problems. Attempts are being made to use grid tools to elicit knowledge for synthesis problems. For example, we are experimenting with allowing the user to put arbitrary computations in the cells of repertory grids in AQUINAS (spreadsheet-like computations, database access calls, functions that sample sensors). However, this will make the tool more difficult to use for these applications.

We are also planning to allow experts to specify complex interacting constraints between traits in AQUINAS. This will help capture knowledge for portions of configuration and scheduling problems.

Eventually, we hope to blend together techniques from other tools (such as MDIS, Antonelli, 1983, and SALT, Marcus, 1985, 1987) to help elicit knowledge for aspects of design problems:

- Acquisition of causal models and design constraints.
- Selecting alternative design concepts based on competing criteria such as reliability, maintainability, cost, manufacturability.
- Acquiring human expertise in validation and verification of designs.
- Acquiring knowledge for design evaluation and change recommendation.
- Acquiring knowledge for document generation and evaluation.
- Capturing historical data in a "corporate memory" database to help solve future design problems.

Other knowledge acquisition tools are useful in producing knowledge-based systems more rapidly than manual interviewing methods. Knowledge acquisition tools may be characterized by the problem tasks for which they are designed to gather knowledge and the problem-solving method that uses the knowledge to solve a problem (Boose, 1988b). Particular knowledge acquisition tools can be viewed as linking tasks and methods.

AQUINAS attempts to capture knowledge for most types of analysis problems, other systems such as MOLE and MDIS derive their power from knowing more about the application domain.

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 Miroslav Benda, Kathleen Bradshaw, Brian Gaines, Cathy Kitto, Ted Kitzmiller, Art Nagai, Doug Schuler, Mildred Shaw, Suzzanne Shema, and Lisle Tinglof-Boose, 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.

Antonelli, D., The Application of Artificial Intelligence to a Maintenance and Diagnostic Information System (MDIS), in the *Proceedings of the Joint Services Workshop on Artificial Intelligence in Maintenance*, Boulder, CO., 1983.

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., Uses of Repertory Grid-Centered Knowledge Acquisition Tools for Knowledge-Based Systems, Special Issue on the Second Knowledge Acquisition for Knowledge-Based Systems Workshop, *International Journal of Man-Machine Studies*, in press, 1988.

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 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1, 1987.

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

Chang, E., Participant Systems, *Future Computing Systems*, Vol. 2, No.1, 1986.

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.

Kass, A., and Leake, D., Types of Explanations, Yale Computer Science Report YALEU/CSD/RR #523, March, 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 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2, 1987a.

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

Kitto, C. M., and Boose, J. H. (1988). Selecting Knowledge Acquisition Tools and Strategies Based on Application Characteristics, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Marcus, S., McDermott, J., and Wang, T., Knowledge Acquisition for Constructive Systems, in the *Proceedings of the Ninth Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 1985.

Marcus, S., Taking Backtracking with a Grain of SALT, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4, 1987.

Saaty, T. L., The Analytic Hierarchy Process, New York: McGraw-Hill, 1980.

Schank, R. C., Explanation Patterns: Understanding Mechanically and Creatively, Lawrence Erlbaum Associates, Englewood Cliffs, NJ, 1986.

Shaw, M. L. G., and Gaines, B. R., Techniques for Knowledge Acquisition and Transfer, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No.1, 1987.

Shema, D. B., and Boose, J. H., Refining Problem-Solving Knowledge in Repertory Grids Using a Consultation Mechanism, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press, 1988.

LEARNING AT DIFFERENT LEVELS OF KNOWLEDGE

YVES KODRATOFF
UA 410 du CNRS
Laboratoire de Recherche
en Informatique, Bat.490
Université de Paris-Sud
91405, Orsay Cedex
FRANCE

GHEORGHE TECUCI
Research Institute
for Computers
and Informatics
71316, Bd.Miciurin 8-10
Sector 1, Bucharest
ROMANIA

Abstract

DICIPLE is a Knowledge Acquisition (KA) system in which Machine Learning (ML) techniques have been integrated. Such an integration raises several questions among which the level of accuracy required from the human user during his/her use of the system. This paper describes DISCIPLE's answer to this question. The learning mechanisms depend on the type of the domain theory (weak or strong). However, a domain is seldom totally weak or totally strong. It may have parts easy to formalize and parts very difficult to formalize. We therefore take the view that a learning apprentice system should be able to learn at different levels of knowledge, in order to take full advantage of the available domain theory. In this paper we present developments of the learning mechanisms of DISCIPLE, enabling it to adapt to the available theory: poor, complete, incomplete, or slightly incorrect.

1. Introduction

Merging KA and ML systems (such systems will be named KAMLS in the following) raises several questions which we believe reduce to three main ones: Should the user have access to the generated knowledge? Should we allow human experts to provide different answers to the same problem? At what level of accuracy (or of formalism) should work both the system and its user?

This paper mainly adresses the last question, but we would like to give brief indications on how DISCIPLE deals with the first two ones.

1.1 - Should the user have access to the generated knowledge?

In classical KA systems, the user provides the rules to be used. In a KAMLS, some generalization of these rules may occur. When this generalization takes place through the replacement of some constants by variables, as it is the case in DISCIPLE, the rules may become quickly unreadable. In the present implementation, DISCIPLE does make available its generalizations to the user, and our experience shows that most users have difficulties tracking the generated variables. The forthcoming implementation will therefore hide completely this feature to its user, and all generalizations will be shown through their instances only, as described in this paper. Notice that such a generalization will usually be somewhat wrong, and will therefore provide instances that will be rejected by the expert, and hopefully some also that will be recognized as valid by the expert. DISCIPLE uses both of them in order to refine its rules, thus achieving an interrogation of its human expert about the validity of variabilized rule, but showing instances only to the human expert.

1.2 - Should we allow human experts to provide different answers to the same problem?

This paper does not deal with this issue. It is nevertheless very important, especially with regards to the fact that DISCIPLE offers the facility (which should be always available on any KA system) to allow different users to work in succession on the same knowldge base. In practise, this means that most of the clerical job can be done by one or several "low-level" experts, while the refined theory will be put

in by a real expert who will concentrate on difficult solutions. Real experts often contradict each other, to say nothing about their behaviour towards less qualified ones. It is of course highly desirable to leave them the freedom of expressing openly their differences, and to make use of these differences to build strategies on how to use the rules. If such a system could be achieved it would be the best way to get meta-knowledge from the experts. Improving DISCIPLE along such lines is part of our current research effort.

1.3 - At what level of accuracy (or of formalism) should work both the system and its user?

To build an expert system one has to face the difficult problem of knowledge acquisition. One recent approach to this problem is to build a learning apprentice system [Mitchell & al. 1985] which is provided with an initial domain theory and is able to develop the knowledge base by interactivly solving problems and assimilating new knowledge by observing, analyzing and questioning about the problem solving steps contributed by its user through his normal use of the system.

Such an approach is also illustrated by our system DISCIPLE [Kodratoff & Tecuci, 1987a].

DISCIPLE was based on the following assumptions:

- most real world domains are so complex that one can only provide a poor theory of these domains;
- even such a poor theory can guide learning.

The approach was successful in that we were able to define a learning method which based its power on the sinergism of different learning paradigms: explanation-based learning [DeJong & Mooney, 1986; Mitchell & al. 1986; Silver, 1986], learning by analogy [Carbonell, 1986; Gentner, 1983; Winston, 1986], similarity-based learning [Kodratoff & Ganascia, 1986; Michalski, 1983; Mitchell, 1978], and learning by asking questions [Sammut & Banerji, 1986].

However, the position adopted in DISCIPLE was an extreme one. A typical real world domain has parts that may be described by a poor theory [Bareiss & Porter 1987; Kodratoff & Tecuci 1987], parts that may be described by a strong theory [DeJong & Mooney, 1986; Mitchell & al. 1986], and part that may be described by a medium-strong theory [Carbonell & Gil 1987; Rajamoney & DeJong 1987].

A learning apprentice system should take advantage of these different levels of knowledge in its domain theory.

In this paper we present developments of the learning mechanisms of DISCIPLE, enabling it to adapt to the available theory:

- when learning a rule in that part of the domain which has a poor theory it will use the learning method presented in [Kodratoff & Tecuci 1987a; Tecuci & al. 1987].
- when learning a rule in that part of the domain which has a strong theory it will use an explanation-based learning method.
- when learning a rule in that part of the domain which has an incomplete (or slightly incorrect) theory, it will use a combination of the above two methods.

We shall illustrate each of the above cases. However, in order to facilitate the understanding of the paper we shall consider the learning of the same rule, each time with a different type of supporting domain theory.

2. The learning environment

The learning mechanisms of DISCIPLE will be illustrated in the context of the following application [Kodratoff & Tecuci 1987b]:

- given the (incomplete) specifications of an object
- design a plan of actions (a technology) for manufacturing an object satisfying the given specifications.

This problem is similar to those solved by NOAH [Sacerdoti, 1975], NONLIN [Tate, 1977], or SIPE [Wilkins, 1984].

The design of a technology is viewed as a successive decomposition of complex actions into simpler ones, and better defining these simpler actions by choosing tools, materials or verifiers, which are in turn successively refined.

A complete theory of the domain would include knowledge of:

- all the objects of the domain, each object being defined by all the relevant factual properties and relations with other objects. These may be explicitly specified or may be deduced by using inference rules.

- all the actions which could be performed, each action model specifying all the relevant preconditions, all the effects and all the objects on which it may be applied.

A third type of knowledge consists of problem reduction rules (decomposition rules and specialization rules). It is precisely this knowledge that DISCIPLE is trying to acquire.

[Mitchell & al. 1986] and [Rajamoney & DeJong, 1987] define different ways in which a theory could be imperfect. Here we shall consider the imperfections resulted from the following facts: objects, inference rules or actions may be missing, may be incompletly specified, or may be erroneously specified.

The learning problem of DISCIPLE may be formulated as follows:
Given:
- a domain theory;
- a problem to solve;
- a partial solution to the problem, in the form of a decomposition or a specialization of it.

Determine:
- a general decomposition or specialization rule.

For instance,
Given:
- the theory of loudspeaker's manufacturing;
- the problem of attaching two parts of the loudspeaker:
  ATTACH ring ON chassis-membrane-assembly
- the decomposition of this problem into two simpler subproblems expressing the gluing of the two parts with an adhesive:
  APPLY mowicoll ON ring
  PRESS ring ON chassis-membrane-assembly

Determine:
- a general decomposition rule indicating the conditions under which one may reduce an ATTACHment problem to a process of gluing:

```
        IF
<condition>
            (TYPE x solid) & (TYPE y solid) & (MATCHES x y) &
            (ISA z adhesive) & (TYPE z fluid) & (GLUES z x) &
            (GLUES z y)
      THEN
solve the problem
            ATTACH x ON y
by solving the subproblems
            APPLY z ON x
            PRESS x ON y
```

Figure 1. A decomposition rule.

In the following sections we shall show how this problem is solved in the context of the theory

types mentioned in introduction.

3. Learning in a poor theory

In the process of building a domain theory for loudspeaker manufacturing we realized that it was very difficult for our expert (Zani Bodnaru from the Industrial Electronic Entreprise in Bucharest) to describe the actions in terms of their preconditions and effects. On the contrary, it was very easy for him to describe the objects and to give us examples of decompositions and specializations.

This should not surprise anyone working in action planning who knows the difficulty of defining action models for complex operations.

Instead of forcing the expert to formalize his knowledge, we decided to accept the theory which was easily provided by him and to learn the rest of the necessary knowledge.

The resulting theory, which we call poor, misses entirely the action models. Moreover, it does not contain all the relevant properties and relations of the objects and even not all the objects of the domain.

A sample of this theory is represented in figure 2.

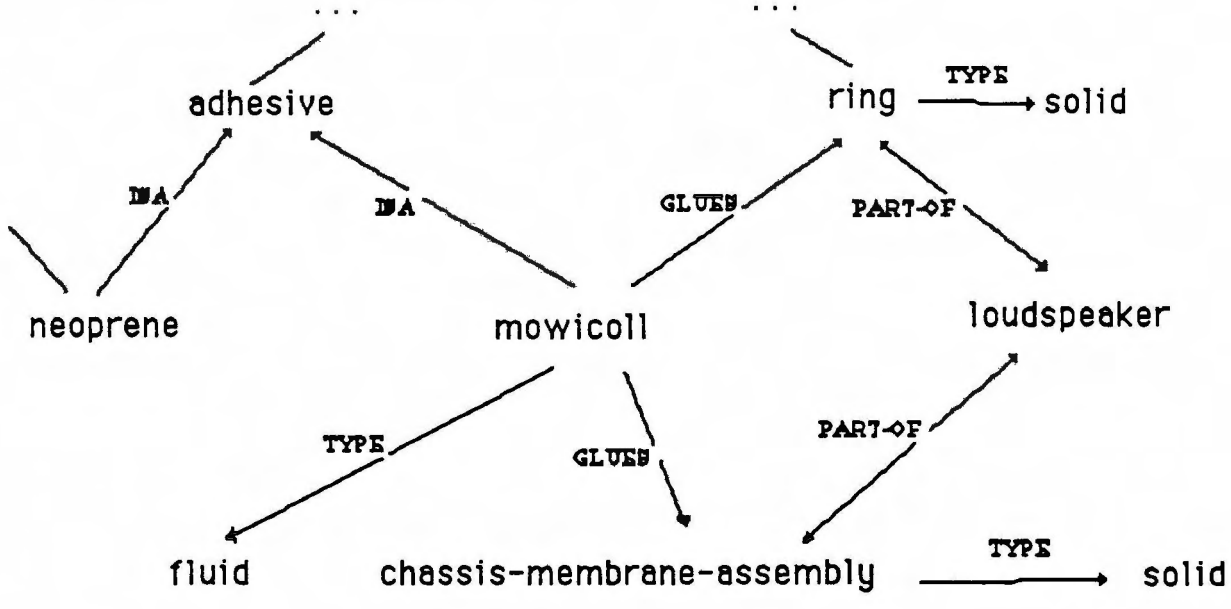


Figure 2. Fragment of a poor theory.

Once this theory is represented into the knowledge base, the system starts problem solving, that is, starts designing plans for manufacturing different types of loudspeakers.

The user formulates the initial problem, for instance:

MANUFACTURE loudspeaker

and the system will try to decompose (or specialize) this problem into simpler subproblems, each subproblem being a new problem to solve. This process continues untill the user obtaines a detailed enough plan for manufacturing the current loudspeaker.

In this process, the system may be unable to decompose (or specialize) some problems. In this case the solution has to be provided by the user, as illustrated in figure 3.

Example 1:
In order to solve the problem
ATTACH ring ON chassis-membrane-assembly
solve the subproblems
APPLY mowicoll ON ring
PRESS ring ON chassis-membrane-assembly

Figure 3. An example of an instantiated rule.

The problem and its solution are considered to be an example of some general rule to be learned.

First DISCIPLE looks for an explanation of user's solution. Then it uses this explanation to formulate a reduced version space [Mitchell, 1978] for the rule to be learned. Each rule in this space covers only instances which are analogous with the user's example. DISCIPLE carefully generates analogous instances to be characterized as positive examples or as negative examples by the user. These are used to further narrow the version space.

In the following we shall intuitively present this learning method. More details and justifications are given in [Kodratoff & Tecuci, 1987a].

In its first learning step, DISCIPLE enters the Explanation Based Mode and tries to find an explanation (within its poor domain theory) of the validity of the solution in figure 3. But the domain theory contains only shallow and/or incomplete knowledge concerning this example. Therefore, the process of finding the explanation is based on the following heuristic: look for an explanation expressible in terms of the relations between the objects from the example ('ring', 'chassis membrane assembly', and 'mowicoll'). To find such an explanation, DISCIPLE will look in its knowledge base for the links and for the paths (i.e. sequences of links) connecting 'ring', 'chassis membrane assembly', and 'mowicoll', and will propose the detected connections as pieces of explanations of the example. It is the user's task to validate them as true explanations:

Do the following justify your solution:
ring PART-OF loudspeaker &
chassis-membrane-assembly PART-OF loudspeaker? No
mowicoll GLUES ring? Yes
mowicoll GLUES chassis-membrane-assembly? Yes

All the pieces of explanations marked by a user's yes form the explanation of the example rule:

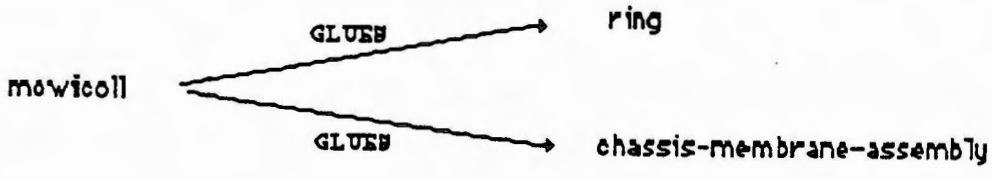


Figure 4. The explanation of Example 1.

Notice that this explanation may be incomplete. This is due partly to the incompleteness of the domain theory and partly to the heuristic used to find explanations (DISCIPLE looks only for the relations between objects, ignoring their properties). However, the detected explanation shows some important features of the objects, features justifying the user's solution.

In its second step, DISCIPLE enters the Analogy-Based Mode. The analogy relies on the observation that similar causes tend to have similar effects and on the fact that, in DISCIPLE, the explanation of a problem solving operation may be regarded as a cause for performing the operation. Therefore, if the network of relations between other objects (for instance, 'screening-cap', 'loudspeaker', and 'neoprené) is similar with Explanation 1, then DISCIPLE may deduce, by analogy with the Example 1,

a new decomposition (see figure 5).

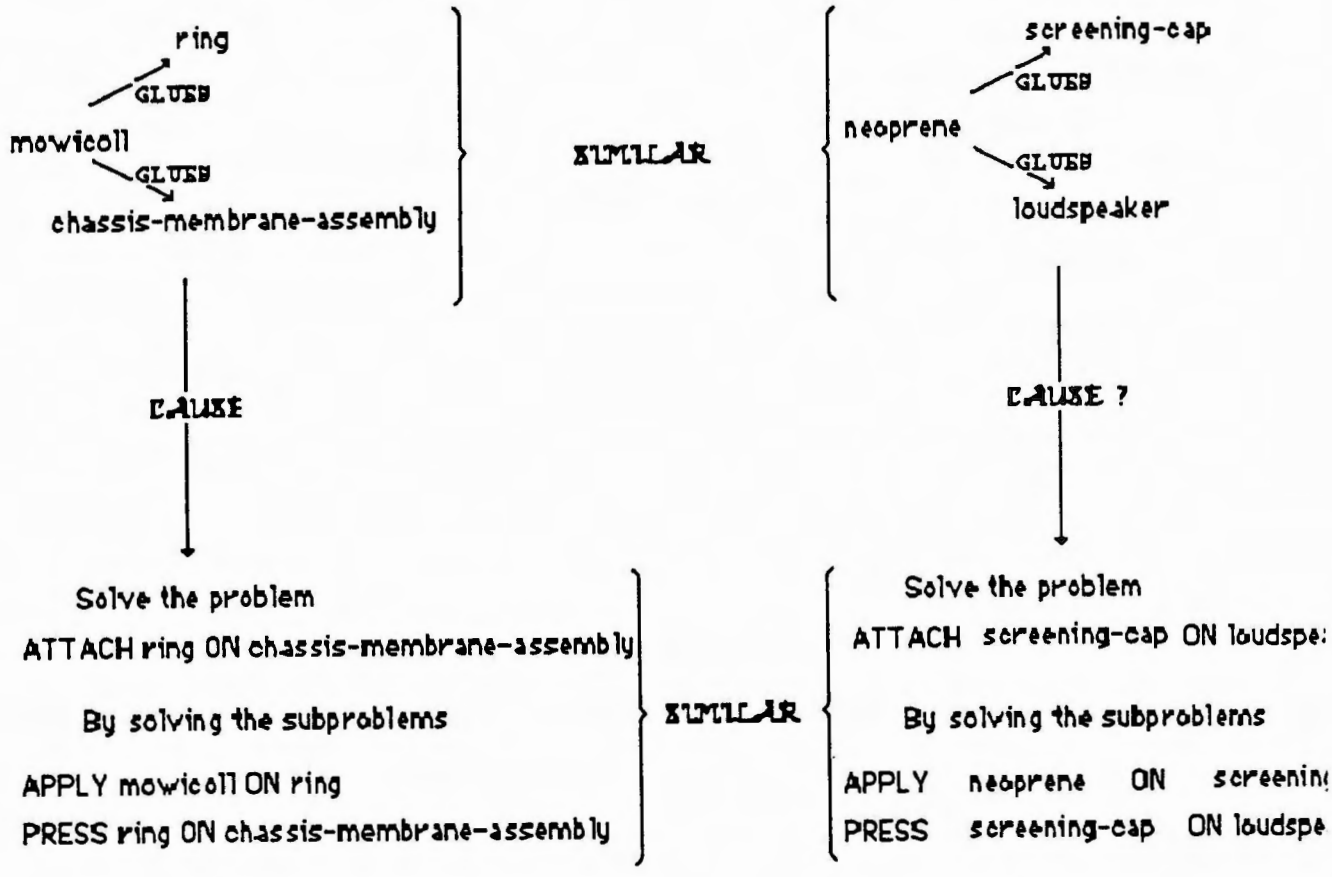


Figure 5. The analogy paradigm in DISCIPLE.

First of all DISCIPLE over-generalizes Example 1 by turning all the objects into variables, thus obtaining:

General Rule 1:
solve the problem
ATTACH x ON y
by solving the subproblems
APPLY z ON x
PRESS x ON y

Next Explanation 1 is rewritten as an (Almost) Sufficient Condition for the application of General Rule 1:

(Almost) Sufficient Condition:
(ISA x ring) & (ISA y chassis-membrane-assembly) &
(ISA z mowicoll) & (GLUES z x) & (GLUES z y)

Notice that the above expression is in fact a true sufficient condition because it reduces General Rule 1 to Example 1 which is known to be true. Nevertheless, as soon as this expression will be generalized to cover new explanations it may no longer be considered a sufficient condition because the explanations may be incomplete.

Further, Explanation 1 is generalized to the most general expression that may still be accepted by

the user as an explanation of General Rule 1. This over-generalization is taken as an upper bound for the applicability condition of General Rule 1:

Almost Necessary Condition:
(GLUES z x) & (GLUES z y)

Figure 6. An over-generalization of Explanation 1.

We have called this over-generalization an Almost Necessary Condition because it is supposed to cover most of the cases of applicability of General Rule 1.

The Almost Sufficient Condition, the Almost Necessary Condition and the General Rule 1 define a reduced version space [Mitchell, 1978] for the rule to be learned:

IF
G:(Almost Necessary Condition)
(GLUES z x) & (GLUES z y)

S:(Almost Sufficient Condition)
(ISA x ring) & (ISA y chassis-membrane-assembly) &
(ISA z mowicoll) & (GLUES z x) & (GLUES z y)
THEN
solve the problem
ATTACH x ON y
by solving the sequence of subproblems
APPLY z ON x
PRESS x ON y

Each rule in this space has an applicability condition that is less general that the Almost Necessary Condition and more general than the Almost Sufficient Condition. Also, it covers only instances that are analogous with Example 1. To search the rule in this space, DISCIPLE needs positive and negative instances of it. These instances may be provided by future problem solving episodes or may be generated by the system itself. To generate such an instance, DISCIPLE looks in the knowledge base for objects satisfying the Almost Necessary Condition. It calls Explanation i the properties of these objects that were used to prove that they satisfy the Almost Necessary Condition:

Explanation i:
(ISA x screening-cap) & (ISA y loudspeaker) &
(ISA z neoprene) & (GLUES z x) & (GLUES z y)

It uses the detected objects to generate an instance of General Rule 1 (see figure 5) and asks the user to validate it:

May I solve the problem
ATTACH screening-cap ON loudspeaker
by solving the subproblems
APPLY neoprene ON screening-cap
PRESS screening-cap ON loudspeaker?

Let us suppose that the user accepts this decomposition. In this case DISCIPLE computes a maximally specific common generalization of the Almost Sufficient Condition and Explanation i:

Explanation i:
(TYPE x solid) & (TYPE y solid) & (ISA z adhesive) &
(GLUES z x) & (GLUES z y)

This generalization is taken as a new Almost Sufficient Condition. Notice that this is always more specific than the Almost Necessary Condition because both Almost Sufficient Condition and Explanation i are less general than the Almost Necessary Condition.

Then, DISCIPLE looks for other objects satisfying the Almost Necessary Condition and generates

another instance of General Rule 1:

May I solve the problem
ATTACH screening-cap ON loudspeaker
by solving the subproblems
APPLY scotch ON screening-cap
PRESS screening-cap ON loudspeaker? No

Figure 7. A negative example of the rule to be learned.

Let us suppose that the user rejects this decomposition. In this case DISCIPLE looks for an explanation of the failure because this explanation points to the important object features which were not contained in Explanation 1. The explanation is that 'scotch' (an adhesive tape) is not fluid (therefore, it might not be applied on a curved surface):

Failure Explanation:
NOT (TYPE scotch fluid)

The negation of this explanation is used to specialize both bounds of the version space.

In another experiment, failing to glue two objects whose surfaces do not match each other, DISCIPLE discovers the condition that the objects should match:

IF
G: (Almost Necessary Condition)
(GLUES z x) & (GLUES z y)
(TYPE z fluid) & (MATCHES x y)

S: (Almost Sufficient Condition)
(TYPE x solid) & (TYPE y solid) &
(ISA z adhesive) & (GLUES z x) & (GLUES z y) &
(TYPE z fluid) & (MATCHES x y)
THEN
solve the problem
ATTACH x ON y
by solving the subproblems
APPLY z ON x
PRESS x ON y

The learning process decreases the distance between the two boundaries of the version space. This process should, in principle, continue untill the upper bound becomes identical with the lower one. Thus one learns the rule in figure 1. However, since the domain theory is poor, we must expect that this will not always happen. Therefore, we will be forced to preserve two conditions (the upper bound and the lower bound), instead of a single applicability condition.

It should be noticed that the degree of generalization of the learned rule is determined by the degree of generalization of the over-generalized explanation in figure 6. However, this rule may still be generalized in the future. This opportunity arises when the user indicates that two objects may be attached by a gluing process even if the rule does not apply. In such a case, the condition(s) of the rule may be generalized, so that to cover the new example as well.

4. Learning in a strong theory

In the following we shall show how the same learning problem is solved in the context of a strong (complete) theory.

The domain theory consists of the following:

- factual properties of objects and relations between objects. They are illustrated in the network from figure 2.

- rules for inferring new properties and relations:
(GLUED x y) --> (ATTACHED x y) & (ISA z adhesive) & (GLUES z x) & (GLUES z y) & (BETWEEN z x y)
(GLUED x y) --> (GLUED y x)
(GLUED x y) --> (ADHERENT-ON x y)

- action models:

Action	Preconditions	Effect
ATTACH x ON y	(TYPE x solid) (TYPE y solid)	(ATTACHED x y)
APPLY z ON x	(TYPE z fluid) (ADHERENT-ON z x) (TYPE x solid)	(APPLIED-ON z x)
PRESS x ON y	(MATCHES x y) (APPLIED-ON z x) (TYPE y solid)	(BETWEEN z x y)

One may notice that the features of the objects on which an action may be applied are specified in the action's preconditions.

In this case, the learning method is a pure explanation based one:
First of all the system proves that the solution indicated by the user is indeed a solution of the problem to solve.
Secondly it generalizes the proof tree as much as possible so that the proof remains valid.
Thirdly it formulates the learned rule from the generalized proof.

Let us consider again the example in figure 3. To prove this example means to show that the (sequence of the) decomposing actions (APPLY, PRESS) achieve the goal of the decomposed action (ATTACH):

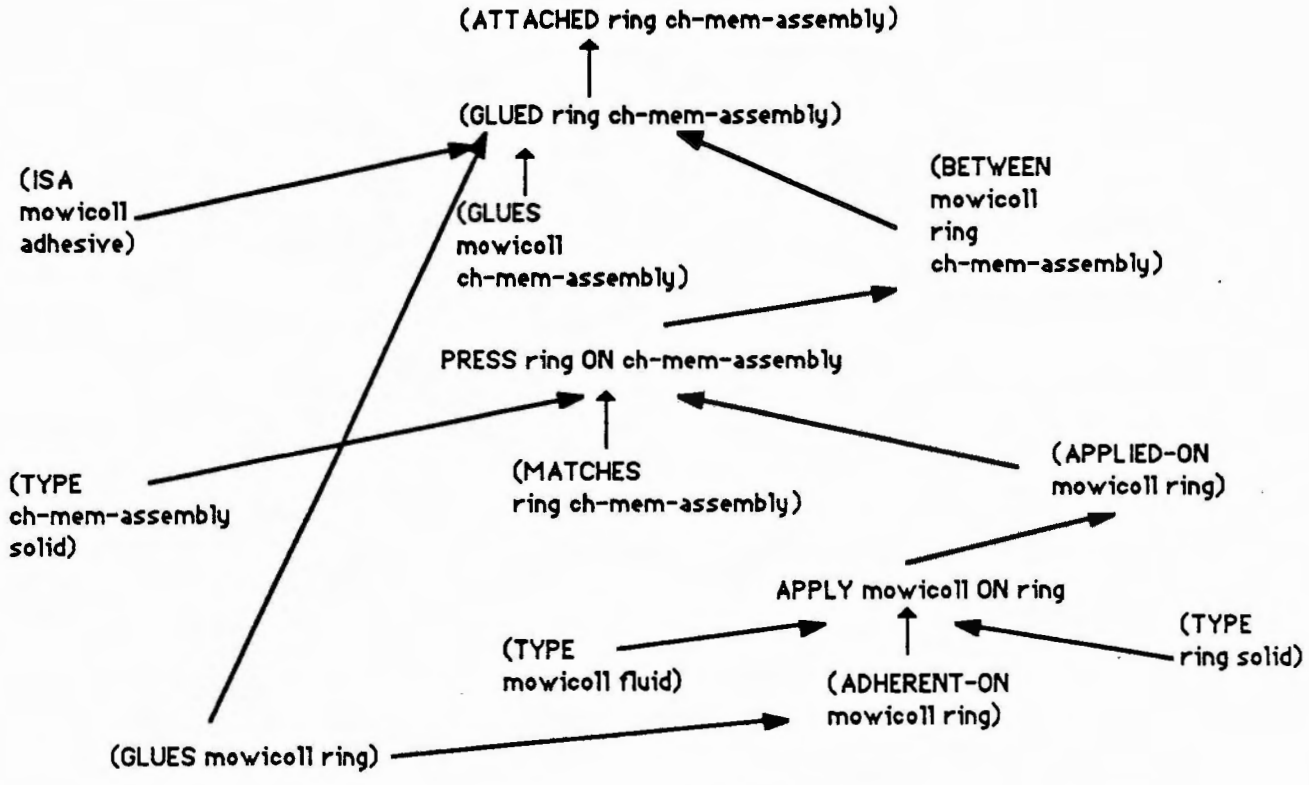
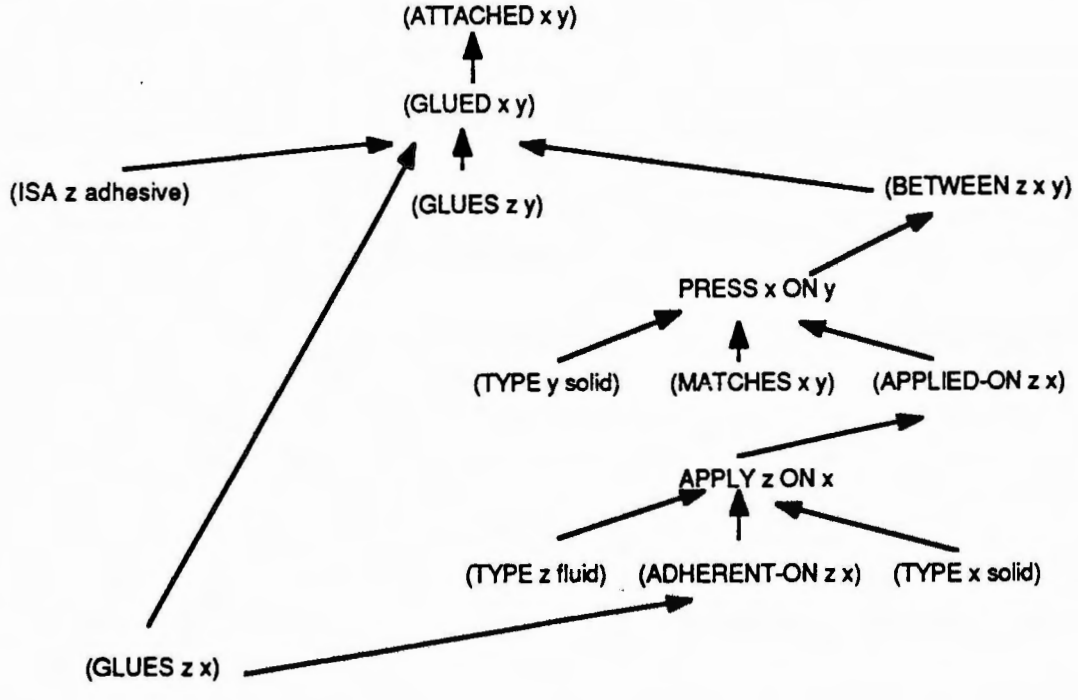


Figure 8. A complete proof of user's solution.

In the above proof, 'ch-mem-assembly' stands for 'chassis-membrane-assembly'.

This proof is generalized by replacing each instantiated inference rule or action model with its general pattern and by unifying the patterns [Mooney & Bennet, 1986]:



The leaves of this generalized tree represent a general precondition for which the sequence of APPLY and PRESS achieves the goal of ATTACH.

This may be expressed in the form of the following general decomposition rule:

IF
    <condition>
    (TYPE x solid) & (TYPE y solid) & (MATCHES x y) &
    (ISA z adhesive) & (TYPE z fluid) &
    (GLUES z x) & (GLUES z y)
THEN
    solve the problem
    ATTACH x ON y
    by solving the subproblems
    APPLY z ON x
    PRESS x ON y

5. Learning in an incomplete theory

Here we shall show how one can deal with an incomplete domain theory defined as follows: some objects, inference rules, and actions are missing or are partially specified.

A partially specified object misses certain properties or relations with other objects, a partially specified action misses some of its precondition predicates or some of its effect predicates, and a partially specified inference rule misses some of its left hand side or right hand side predicates.

In this case the learning method combines the two learning methods presented previously:

First of all, the system will construct an incomplete proof of the user's solution and will generalize it (as in a complete theory).

Secondly the system will start experimentation in order to find an exact explanation (as in a poor theory).

The first step may require asking foused questions to the user, in order to fill the possible gaps in the proof.

Also, the proof found in the first step will provide additional focus for the second step.

As a side effect of rule learning, one will develop the domain theory.

Even when the objects, the inference rules, and the actions are incompletly specified one may be able to construct a proof tree which misses some of the portions of the complete proof tree. When the system misses inference rules or action models it will try to sketch the proof tree both from top-down and from bottom-up and will ask the user focused questions in order to connect the different parts of the proof.

To illustrate this, let us suppose that the domain theory misses the model of the PRESS action and that the model of the APPLY action misses the precondition predicate (TYPE z fluid). In this case the system was able to build the following proof structure for the example in figure 3:

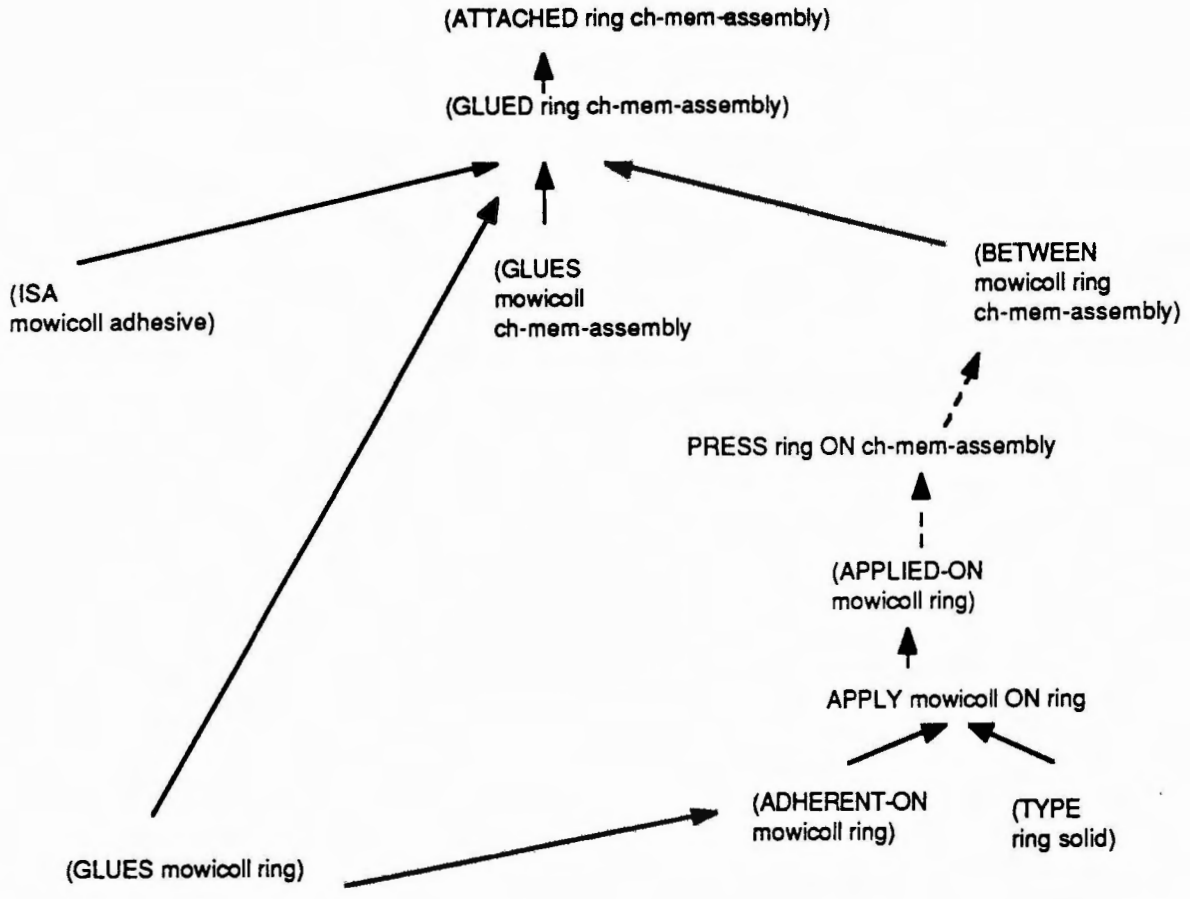


Figure 9. An incomplete proof of user's solution.

The dotted lines from the above proof structure do not result from the domain theory but are hypothesis made by the system and confirmed by the user.

For instance, the system makes the hypothesis that

(BETWEEN mowicoll ring ch-mem-assembly)

is an effect of the action

PRESS ring ON ch-mem-assembly

from the fact that all the other left hand side predicates of the inference rule

(ISA z adhesive) & (GLUES z x) & (GLUES z y) & (BETWEEN z x y)
--> (GLUED x y)

are already true in the initial situation and the predicate 'BETWEEN' is not true.

Comparing this proof tree with the one in figure 8, one may easily notice that it misses some of the portions of the complete proof tree. Nevertheless, its leaves provide the following (incomplete) explanation for the example in figure 3:

Explanation:
(ISA mowicoll adhesive) & (GLUES mowicoll ring) &
(GLUES mowicoll ch-mem-assembly) & (TYPE ring solid)

Another effect of this process consists in defining an initial model for the action PRESS:

Action	Preconditions		Effects
PRESS x ON y	lower bound: (APPLIED-ON z x) (ISA x ring) (ISA y ch-mem-assembly) (ISA z mowicoll)	upper bound: (APPLIED-ON z x)	(BETWEEN z x y)

During the learning of the decomposition rule in figure 1, the system will also refine the model of the PRESS action.

Once the proof in figure 9 is built, the system will generalize it, as in a complete theory:

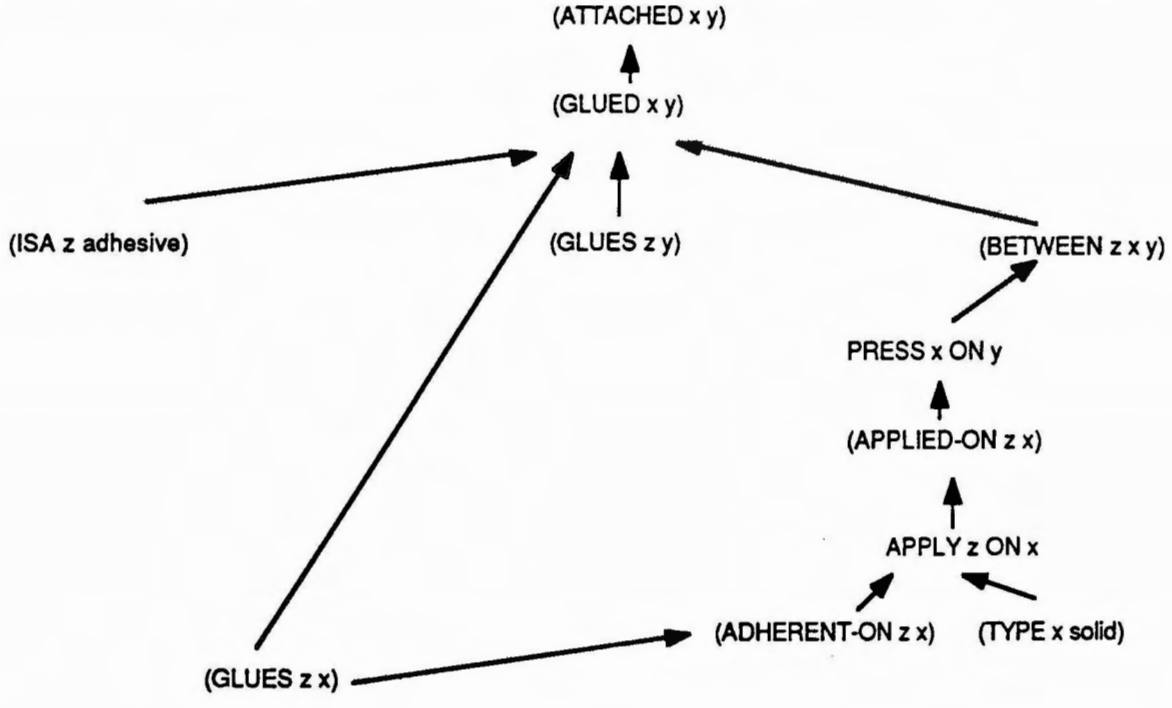


Figure 10. The generalization of the proof in figure 9.

The leaves of this generalized proof tree provide an over-generalized explanation for user's solution:

Over-generalized explanation:
(ISA z adhesive) & (GLUES z x) & (GLUES z y) &
(TYPE x solid)

Therefore, the system is able to formulate the following intermediate rule (as in a poor theory):

IF

upper bound

(ISA z adhesive) & (GLUES z x) & (GLUES z y) &
(TYPE x solid)

lower bound

(ISA x ring) & (TYPE x solid) &
(ISA y ch-mem-assembly) &
(ISA z adhesive) & (GLUES z x) & (GLUES z y)

THEN

solve the problem

ATTACH x ON y

by solving the subproblems

APPLY z ON x
PRESS x ON y

Figure 11. An intermediate form of the rule to be learned.

This rule will be improved during future experiments, as in a poor theory. This time, however, the generalized proof tree in figure 10 provides a focus for the process of finding the explanations of the failures. To illustrate this, let us consider again the failure in figure 7. In this case, the system generates the instance of the generalized proof in figure 10, corresponding to the problem solving episode in figure 7 (by replacing 'x', 'y', and 'z' with 'screening-cap', 'loudspeaker', and 'scotch', respectively). The fact that the user rejected the solution proposed by the system proves that the leaves of the instantiated tree do not imply the top of the tree (the leaf predicates are true but the top predicate is not). This means that some action models or inference rules are faulty. To detect them, the system follows the instantiated proof tree from the bottom to the top. If the effect of an action or the consequent of an inference rule is not true than the corresponding action model (inference rule) may be the incomplete one.

In our case, the effect of

APPLY scotch ON screening-cap

is not achieved. This means that 'APPLY' has a precondition which is not contained in its model and is not true in the current situation. This precondition is

(TYPE z fluid)

and it is not satisfied in the case of the problem solving episode in figure 7 because 'scotch' is not fluid:

NOT(TYPE scotch fluid).

The explanation of the failure in figure 7 is precisely the above fact.

Therefore, in an incomplete theory, finding the explanation of a failure reduces to finding the missing knowledge from a knowledge piece (action model or inference rule). In this case, the generalized proof in figure 10 plays the role of a justification structure for the rule to be learned [Smith & al. 1985].

As it was also mentioned in the end of section 3, the generality of the learned rule is limited by the generality of the over-generalized explanation. However, the rule may be further generalized in response to a problem solving situation in which the rule does not apply and the user says that it should apply. In this case, the condition of the rule and some action models or inference rules from the associated generalized proof may be generalized to cover the new situation as well.

6. Learning in an imperfect theory

This section presents a future direction of our research: learning in an imperfect theory.

A theory may be imperfect from different points of view [Mitchell & al. 86; Rajamoney & DeJong, 1987]. However, we shall consider only imperfections resulted from missing knowledge or minor erroneous knowledge in the domain theory. By this we mean that certain pieces of knowledge (objects, inference rules, action models) are missing or contain minor errors in their definitions in that parts of these definitions may be either more general or less general than they should be.

For instance, the preconditions of an action may be:

- more general than they should be, allowing the application of the action in situations in which they should not;
- less general than they should be, forbiding the application of the action in situations in which they should not;
- may have some parts which are more general and other parts which are less general than they should be, having both of the above consequences.

The same types of errors may manifest in the effects of an action, in any of the sides of an inference rule, or in the definition of an object property.

Let us notice that the incomplete theory is a special case of the above defined imperfect theory. For instance, an action missing some precondition predicates has a precondition that is more general than it should be. In particular, an action having no precondition is supposed to be applicable in any situation. Therefore, learning in an imperfect theory may be similar to learning in an incomplete theory.

Besides learning problem reduction rules, the system will also correct the theory. This will consist in appropriate generalizations and particularizations of parts of the knowledge pieces, in response to encountered failures.

7. Discussion and conclusion

In this paper we have shown how a system may adapt its learning method to the available domain theory.

The justification of this approach is that a domain theory is seldom totally weak or totally strong. It may have parts easy to formalize and parts very difficult to formalize. Even if this is not a feature of the domain, its theory may result as such because any human expert knows certain parts of his domain better than others.

We therefore consider that a learning apprentice system should be able to learn with different levels of knowledge, in order to take full advantage of the available domain theory.

It is also interesting to notice that what is learned depends on the type of the domain theory. In a complete theory, learning does not improve the competence of the system but only its performance. The rule to ATTACH two objects by gluing them with an appropriate adhesive provides a ready-made solution to a problem that may anyway be solved by a state space search [Nilsson 1971].

In a poor, incomplete, or incorrect theory, learning improves not only the performance of the system but also its competence. For instance, the system may acquire new action models, enabling it to solve problems behinds its initial capabilities.

In our approach the system performs active experimentation, by generating itself analogous problem solving episodes, in order to improve the rule synthesized after the explanation-based learning phase. This is not essential. The system may keep the incompletly learned rules, testing them on future problem solving episodes. In fact this will anyway happen since the system will not always be able to completly learn a rule.

It is also important to stress how much our approach is based on interaction. We believe that interaction (with a human [Buntine & al. 1987], or with an environment [Carbonell & Gil 1987; Rajamoney & DeJong 1987]) plays a major role in real-world learning systems.

References

[Bareiss & Porter 1987] Bareiss E. and Porter B., PROTOS: An Exemplar-Based Learning Apprentice, in Langley P.(ed) Proc. 4th Int. Workshop on MACHINE LEARNING, Irvine, 1987.

[Buntine & Stirling 1987] Buntine W. and Stirling D., Interactive Induction, 13pg, September 1987.

[Carbonell, 1986] Carbonell J., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition, in Michalski R., Carbonell J. si Mitchell T. (eds) Machine Learning: An Artificial Intelligence Approach, Vol. 2, Morgan Kaufmann 1986, pp. 371-392.

[Carbonell & Gil 1987] Carbonell J. and Gil Y., Learning by Experimentation, in Langley P.(ed) Proc. 4th Int. Workshop on MACHINE LEARNING, Irvine, 1987.

[DeJong & Mooney, 1986] DeJong G., Mooney R., Explanation-Based Learning: An Alternative View, Machine Learning 1, pp. 145-176.

[Gentner, 1983] Gentner D., Structure-Mapping: a Theorectical Framework for Analogy, Cognitive Science, 7, PP. 155-170.

[Kodratoff & Ganascia, 1986] Kodratoff Y. and Ganascia J-G., Improving the Generalization Step in Learning, in Michalski R., Carbonell J. si Mitchell T. (eds) Machine Learning: An Artificial Intelligence Approach, Vol. 2, Morgan Kaufmann 1986, pp. 215-244.

[Kodratoff & Tecuci, 1987a] Kodratoff Y. and Tecuci G., DISCIPLE: An Integrated Expert and Learning System for Weak Theory Fields, LRI Research Report 371, 40 pg, Orsay, September 1987.

[Kodratoff & Tecuci, 1987b] Kodratoff Y. and Tecuci G., Techniques of Design and DISCIPLE Learning Apprentice, International Journal of Expert Systems: Research and Applications, vol.1, nr.1, pp. 39-66, 1987.

[Michalski, 1983] Michalski R., A Theory and a Methodology of Inductive Learning, Artificial Intelligence 20(1983), pp 111-161.

[Mitchell 1978] Mitchell T.M., Version Spaces: An Approach to Concept Learning, Doctoral dissertation, Stanford University.

[Mitchell & al. 1985] Mitchell T., Mahadevan S. and Steinberg L., LEAP: a Learning Apprentice System for VLSI Design, Proc. IJCAI-85, Los Angeles, pp. 573-580.

[Mitchell & al. 1986] Mitchell T.M., Keller R.M., Kedar-Cabelli S.T., Explanation-Based Generalization: A Unifying View, Machine Learning 1, pp. 47-80, 1986.

[Mooney & Bennet, 86] Mooney R. and Bennett S., A Domain Independent Explanation-Based Generalizer, Proc. AAAI-86, Philadelphia, pp. 551-555.

[Nilsson 1971] Nilsson N., Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

[Rajamoney & DeJong, 1987] Rajamoney S. & DeJong G, The Classification, Detection and Handling of Imperfect Theory Problems, Proc. IJCAI-87, Milano, pp. 205-207.

[Sacerdoti 1975] Sacerdoti E., A Structure for Plans and Behavior, Technical Note 109, SRI

International, August 1975.

[Sammut & Banerji, 1986] Sammut C., Banerji R.B., Learning Concepts by Asking Questions, in Machine Learning: An Artificial Intelligence Approach, Volume 2, Michalski R.S., Carbonell J.G., Mitchell T.M. (eds), Morgan-Kaufmann 1986, pp. 167-191.

[Silver 1986] Silver B., Precondition Analysis: Learning Control Information, in Machine Learning: An Artificial Intelligence Approach, Volume 2, Michalski R.S., Carbonell J.G., Mitchell T.M. (eds), Morgan-Kaufmann 1986.

[Smith & al. 85] Smith R., Winston H., Mitchell T. and Buchanan B., Representation and Use of Explicit Justifications for Knowledge Base Refinement, Proc. IJCAI-85, Los Angeles.

[Tate, 77] Tate A., Generating Project Networks, Proc. IJCAI-77, Massachusetts, pp. 888-893.

[Tecuci & al. 1987] Tecuci G., Kodratoff Y., Bodnaru Z. and Brunet T.: DISCIPLE: An Expert and Learning System, in D. S. Moralee (ed): Research and Development in Expert Systems IV, Cambridge University Press, 1987.

[Wilkins, 1984] Wilkins D., Domain Independent Planning: Representation and Plan Generation, Artificial Intelligence 22, pp 269-301.

[Winston, 1986] Winston P.H., Learning by Augmenting Rules and Accumulating Censors, in Machine Learning: An Artificial Intelligence Approach, Volume 2, Michalski R.S., Carbonell J.G., Mitchell T.M. (eds), Morgan-Kaufmann 1986, pp. 167-191.

KRITON

A Knowledge Elicitation Tool For Expert Systems

Marc Linster
Forschungsgruppe Expertensysteme
Institut für Angewandte Informationstechnik
GMD
Postfach 1240
D-5205 Birlinghoven
e-mail : linster@gmdzi.uucp

Abstract

We present the GMD´s knowledge acquisition project, its motivation and a first laboratory prototypical elicitation tool. We discuss the methods integrated, their drawbacks and limits, we explain the systems architecture and show where in the knowledge engineering process such a tool may be used.

Introduction

The efficient and wide-spread use of expert-systems is restrained by the lack of methods and theories on knowledge acquisition. In the expert system group´s research project KRITON at the GMD we try to give some systematic support to this critical phase in the expert system life cycle.

The main questions KRITON should help to answer are :

- how can knowledge be elicited from experts ?

- how can we avoid the knowledge engineer´s bias ?

- which methods allow us to give systematic support to the expert system life cycle ?

- how can we keep control of large knowledge bases ?

Expert systems should not only have some model of the problem domain, but they require a model of the experts problem solving process too. This model must be more than just a superficial representation of input-output-pairs. It must help answer questions about the motivation of problem solving steps to make the results of the system credible (Steels 1987).

This implies that the knowledge acquisition process can not be limited to the acquisition of bits and pieces of knowledge, but that knowledge about the use of the bits is required to really solve problems.

Thus the knowledge engineering process focuses on the construction of an adequate model of the experts problem solving process. To do so the acquisition process can be divided into elicitation, concerned with the gathering of the bits and the pieces, and the

representation part, concerned with the assembly of the bits into problem solving models. These phases are distinct but cyclic. For greater detail of this phase-model see (Christaller et all 1988).

To gather the bits, the pieces and the model the knowledge engineer tries to make as many of the expert´s knowledge sources accessible to the system as possible. Manuals and background literature are analyzed, interviews are conducted, and the expert is surveyed on the task.

The system KRITON

KRITON[1] is a first answer to some of the questions mentioned above. The knowledge elicitation system KRITON supports the knowledge engineer on the job by partially automating some of the tasks described above. This not only guaranties an increase in speed and methodics, but it prevents the knowledge engineer from misinterpreting acquired data, before it is put into a controllable written form. This is because the expert directly interacts with the system, in his words and formulations. The second important point is the neutrality of the interaction, i.e. questions put to the expert contain no preferences or tendencies. "Was the speed of the car too high ?" would be the tendentious version of the inquiry about a car´s speed.

As KRITON asks neutral questions, writes down the experts answers using his words and formulations, it generates a non-operational version of the knowledge base that is accessible to the untrained expert. Its content can be used for tracing and explanation facilities. The knowledge base must be considered non-operational as no knowledge about the use of knowledge to solve a problem has been included.

The methods in KRITON

KRITON integrates a set of knowledge acquisition modules, whose task it is to acquire knowledge from their special domain.

Protocol analysis

Procedural knowledge, i.e. expertise in action is acquired in KRITON´s protocol analysis component. Protocol analysis (Ericsson, Simon 1984) allows to transform concurrent protocols into rules representing the protocol´s procedural and associative aspects.

In KRITON protocols are fed in together with the pauses of speech used to indicate coherent segments.which are transformed into operator-argument-structures. Operator-argument structures consist of a verb together with the nouns this verb refers upon, or of an adjective together with the noun refined through the adjective. We suppose that this is an adequate way to formalize most facets of protocol fragments. The first kind represents actions or changes, whereas the second form represents states and descriptions. These operator-argument structures can then be combined by the expert into rules, thus formalizing associative and procedural aspects of the protocol.

Interview

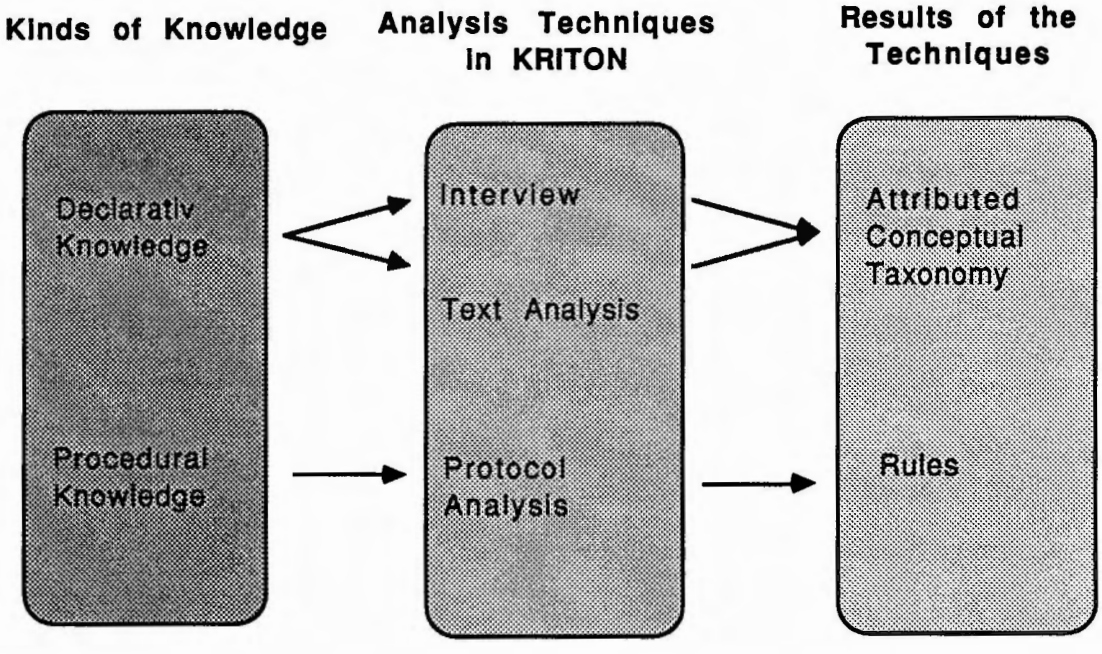
The expert´s knowledge about concepts, their organization and his personal view of the problem domain, i.e. his ways to order and relate the concepts is elucidated using interview techniques strongly influenced by cognitive science (Diederich, Ruhmann, May 1987). Repertory-grid-techniques (Kelly 1955, Boose 1986) are used together with laddering, explicit inheritance and generalization to provide both a describing and

[1]KRITON is named after a disciple of Sokrates

differentiating attribution and a taxonomy of the concepts of the domain. This is supported with a graphical component that allows the expert to arrange concepts the way he sees them, without having to theorize his knowledge to fit some abstract representation formalism that enforces it's view upon the expert, i.e. he does not need to create is-a-hierarchies in the way KL-ONE-like formalisms (Brachmann 1979) sees them, but he can express himself freely.

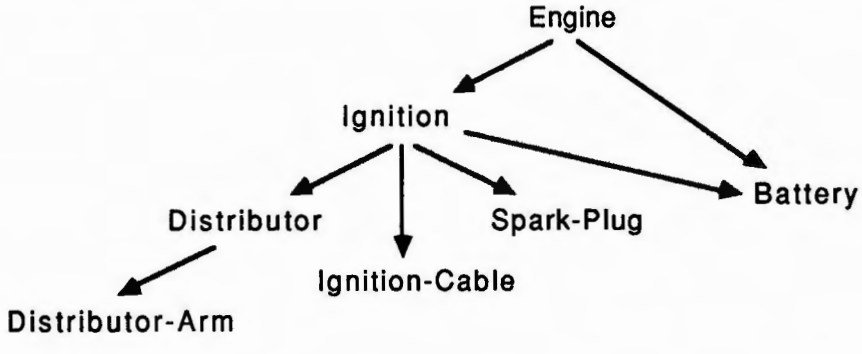
Text analysis

To make sure that the expert is being confronted with as many aspects of the domain as possible, and to reach a pre-structuring of these concepts, KRITON retrieves concepts from texts and allows the user to organize them into taxonomies. This is supported by the same graphical environment as the interview technique.



Illustration[1]

In the text analysis the expert picked the concepts engine, spark-plug, battery, ignition-cable, distributor, distributor-arm out of a text on four-stroke-engines, and roughly organized them according to his view of the domain.



[1]We do not include screen-copies as KRITON is geared towards German language, and it is utterly unable to analyze non-German protocols or texts.

In the interview KRITON questions the expert on the concepts used :

KRITON : What is the difference between spark-plug and ignition-cable ?

Expert :

Attribute : material

Value for spark-plug : ceramics

Value for ignition-cable : copper.

KRITON : What do distributor-arm and distributor-cable have in common, that distinguishes them from battery ?

Expert :

Attribute : voltage

Value for distributor-arm and ignition-cable : high

Value for battery : low

In the protocol analysis component the records of a protocol are analyzed :

...** the car does not start ** there is a spark ** the ignition-cable is wet ** the distributor-arm shows signs of corrosion ** I dry the ignition system ** ...

transformed into operator-argument structures :

(not-start D1[1] car) (present D2 spark) (wet D3 ignition-cable) (corroded D4 distributor-arm) (dry D5 ignition-system)

post-edited and combined into rules by the expert :

IF
$and
(not-start D1 car)
(present D2 spark)
(wet D3 ignition-cable)

THEN
(dry D5 ignition-system)

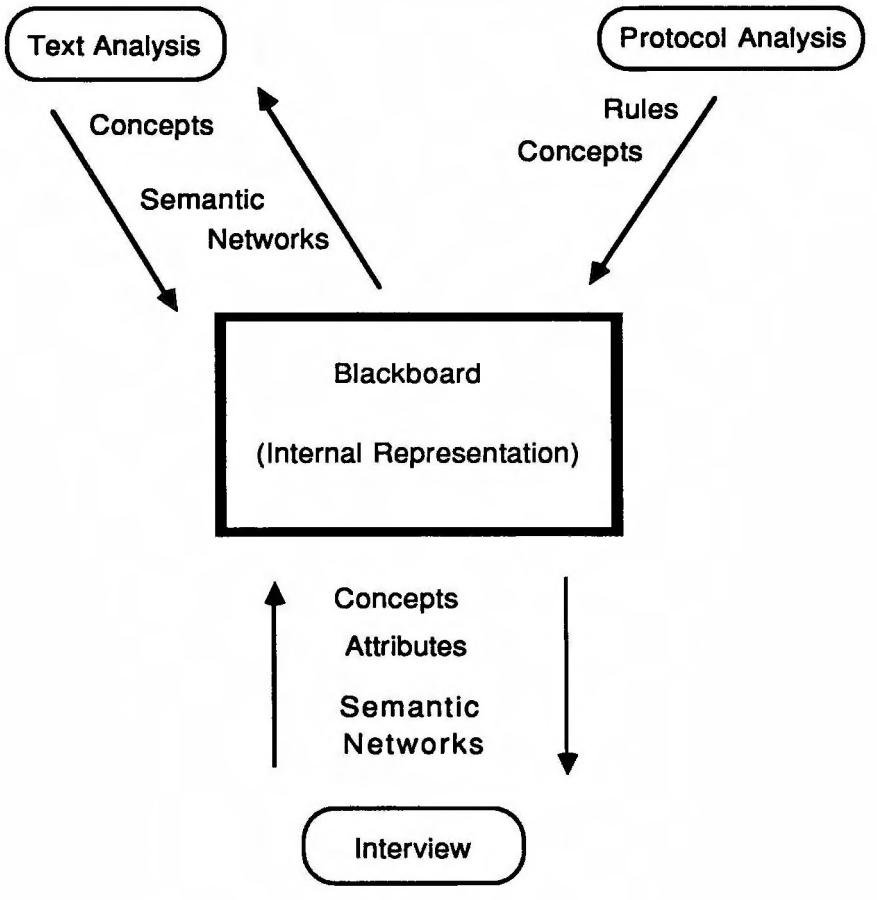
The system´s architecture

KRITON is not just a conglomeration of single techniques, but it organizes them in order to form a symbiosis. The interview finds out what the elements of the domain are, the text analysis tries to bring in as many elements as possible and the protocol analysis sees how these elements are used in the problem solving process. KRITON unites the acquisition modules and enforces a tight cooperation between them to guarantee the

[1]These reference-markers point to the sub-sentence in the actual protocol, that was used to generate the operator-argument structure. They serve documentation purposes.

coherence of the data gathered. Concepts used in rules trigger interviews to integrate them into the conceptual network, concepts appearing in the conceptual network that are not part of any rule can be used to trigger new protocols. For an extensive presentation of this method called symptom generation see (Linster 1987).

The methods interact in a blackboard architecture, aiming at the completion of a common data-structure.



This architecture simplifies the introduction of new techniques as modules. Potential candidates are : networking (Bonato 1988), cluster analysis (Cooke, MacDonald 1986), hypertext (Gaines, Vickers 1987) and consistency checks.

Assessment of the techniques

The techniques integrated in KRITON cannot replace the human knowledge engineer in the development of knowledge based systems, their task has to be limited to supporting the process of knowledge elicitation. The reasons are the computer´s lack of both common sense and all-round education, just as the need for an uninterpreted version of the knowledge base´s contents.

The text analysis cannot capture the full meaning of a document, especially not what is written between the lines. That´s why in KRITON the text analysis proposes elements of the text to the expert who then combines his choice into a conceptual structure.

Similar reasons limit the automated protocol analysis to those tasks not requiring interpretation of the protocols, i.e. the transformation of the segments delimited by pauses of speech into operator-argument structures is wholly automated, whereas the generation of rules is just a computer-supported phase. It must be pointed out that concurrent thinking-aloud protocols, as they are currently processed in KRITON, are only one kind of protocols. Many other techniques, like structured interviewing or concurrent protocols with interruptions by the knowledge engineer, are used in the knowledge engineering process, but can not be processed by KRITON yet.

The interview component uses the conceptual structures generated in the text analysis to ask questions about differentiating attribution, similarities, inheritance, missing concepts, etc. In the interview the potential power of personal construct theory is not extensively used, as the constructs (i.e. the attributes and their values) are only used as descriptions, and are not subject of further inquiries or analysis as proposed in (Gaines, Shaw 1981). It is true too that questioning techniques as they are used in KRITON, or in comparable tools, do not yet reach the power of techniques like the knowledge-acquisition-grid (LaFrance 1986), as the techniques seem too inflexible to cope with the many facets of human expertise.

This careful assessment of KRITON is needed to make sure that these automated techniques can be used appropriately, thus introducing them at those points where they can help increase the performance of the knowledge engineering process through their special properties, e.g. neutrality of interaction, documentation in the experts terminology.

So the protocol analysis is a worthy technique if used to formalize the associative and procedural aspects implicitly contained in the thinking-aloud protocol, but not if used to generate a rule base.

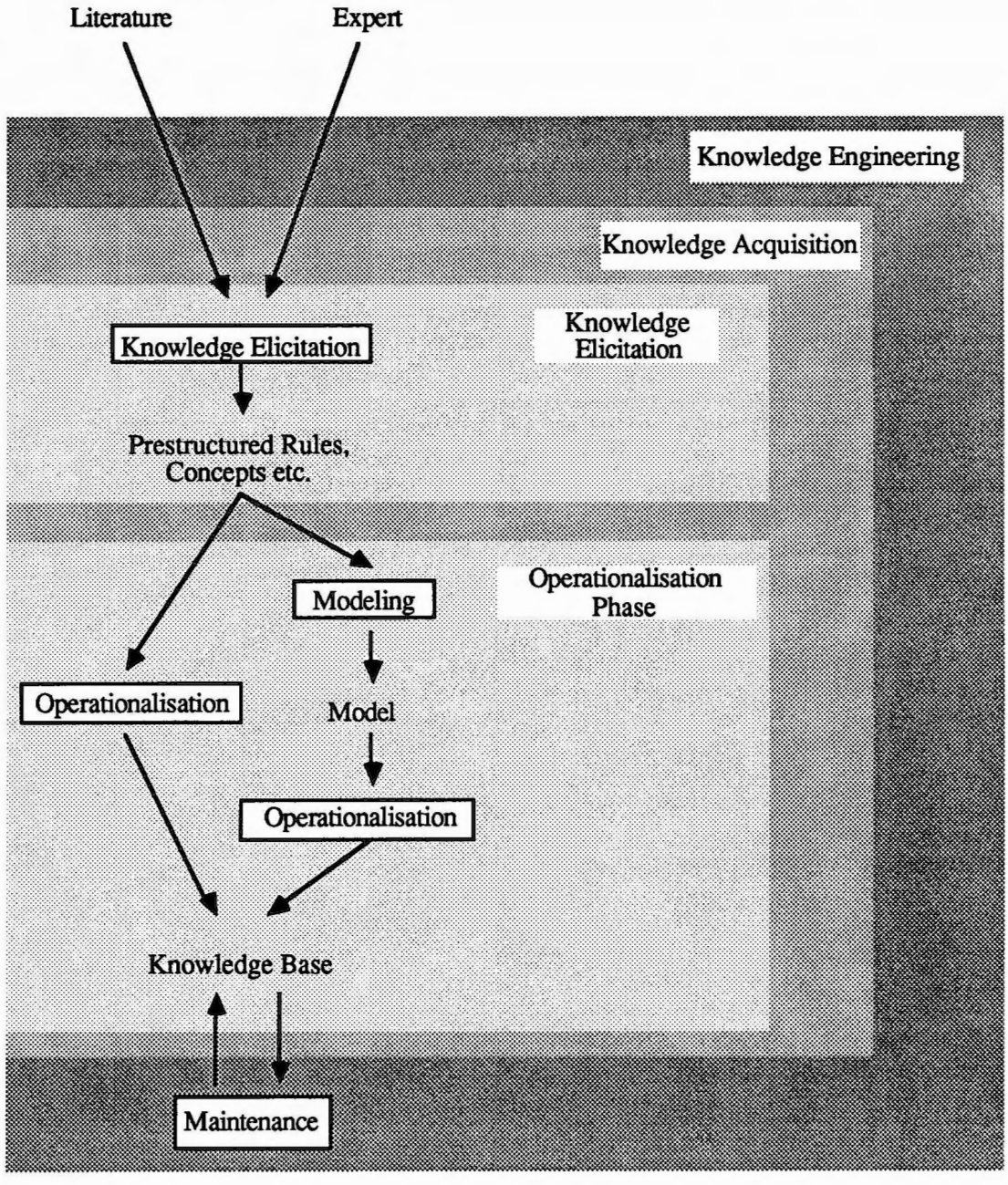
The interview realized in KRITON is a useful technique for the elicitation of concepts and personal constructs but its results, internally represented as attribute-value pairs describing concepts, should not be confounded with an operational frame-base for an expert-system.

This means that our interview, protocol- and text-analysis techniques can very well be used to start-off an elicitation process, but that, from a certain moment on, more powerful ones must take over, using the results from the KRITON-techniques.

Using KRITON

KRITON was conceived to support the development of large systems, where the complexity of the problem, and the solution required, demand for systematic approaches to allow for life cycle support, maintenance or verification.

The systematic approach divides the knowledge acquisition into a knowledge elicitation phase and a design or encoding phase. The results of the elicitation are pre-structured rules and concepts, that in the design phase undergo a modeling process to be encoded into some implementation formalism.



The output of KRITON is a network as dense and complete as possible consisting of concepts, attributes describing the concepts and rules about the concepts. This cannot be considered an operational knowledge base as concepts may be defined unequivocally, synonymous definitions are possible and rules may conflict with hierarchical relations. A modeling phase must attend to these problems.

KADS (Breuker, Wielinga et all 1987) would be a potential modeling methodology KRITON could be integrated into, to transform the results of the elicitation phase into operational models.

Technical Information

KRITON has been implemented on Symbolics Lisp Machines and uses some 500K Bytes. Currently some modules are being implemented on Macintosh, to allow on-site interviewing.

Future Work

As mentioned in the introduction KRITON is a tool that supports the knowledge engineer in the elicitation phase only, and as said in the chapter *Assessment of the Techniques* not the entire elicitation phase is covered. This implies future developments :

- integration of other techniques to cover larger areas of the elicitation domain,

- study of integration aspects, to handle different approaches to the same "bit of knowledge", and to be able to coerce these different approaches into one consistent piece,

- editing-facilities and knowledge-organizing-facilities (Abrett, Burstein 1987, Esfahani, Teski 1987) to cover the representation aspects of the knowledge engineering process. This will allow to include the prototyping phase into the computer-aided knowledge engineering.

Literature

G. Abrett, M. Burstein
The KREME knowledge editing environment
in : B. Gaines, D. Hill (eds.) : International Journal of Man-Machine Studies (1987) 27
London, United Kingdom

M. Bonato
Knowledge elicitation with structure formation techniques
in : J. Diederich, T. Uthmann (eds.); Knowledge Acquisition
Arbeitspapier 281 der GMD, St. Augustin, FRG1988

J. Boose
Expertise Transfer for Expert System Design
Elsevier, New York USA1986

R. Brachmann
On the Epistemological Status of Semantic Networks
In : Findler, N.V. (Ed.): Associative Networks.
Academic Press, New York, USA1979

J. Breuker, B. Wielinga, M van Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, J.-P. Billaut
Model-driven knowledge acquisition : interpretation models
Esprit Report Project 1098
Amsterdam, The Netherlands 1987

N. Cooke, J.McDonald
The application of psychological scaling techniques to knowledge elicitation for expert systems
in : J. Boose, B. Gaines (eds.); Proceedings of "Knowledge Acquisition for Expert Systems"
Banff, Canada 1986

T. Christaller, H.-W. Güsgen, J. Hertzberg, M. Linster, A. Voß, H. Voß
Was ist Expertise und wie bekommt man sie auf den Rechner?
In : Jahresbericht der GMD 1987, St. Augustin, FRG

J. Diederich
Knowledge-Based Knowledge Elicitation
in : J. McDermott (ed.); Proceedings of IJCAI 1987
Morgan Kaufmann Publishers, Los Altos, USA

J. Diederich, I. Ruhmann, M. May
A Knowledge Acquisition Tool for Expert Systems
in : International Journal of Man-Machine Studies (1987) 26
London, United Kingdom

K. A. Ericsson, H.A. Simon
Protocol Analysis
The MIT Press, Cambridge, USA 1984

L. Esfahani, F. Teski
KET, a knowledge encoding tool
In : Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge-Based Systems
University of Reading,U.K.

B.Gaines, M. Shaw
New directions in the analysis and interactive elicitation of personal construct systems
in : M. Shaw(ed.) : Recent Advances in Personal Construct Technology; Academic Press, London 1981

B. Gaines, J. Vickers
Instantiation of Knowledge Structures in Books and Hypermedia
Technical Report; KSI/NML-87-BHM
University of Calgary, Canada

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

M. LaFrance
The knowledge Acquisition Grid
in : J. Boose, B. Gaines (eds.); Proceedings of "Knowledge Acquisition for Expert Systems"
Banff, Canada 1986

M. Linster
On Structuring Knowledge for Incremental Acquisition
In : Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge-Based Systems
University of Reading, U.K.

L. Steels
Second Generation Expert Systems
in : Fachtagung "Expertensysteme 1987", German Chapter of the ACM, Teubner Verlag, FRG 1987

Problems of Validation in a Knowledge Acquisition System using Multiple Experts

Mildred L. G. Shaw
Knowledge Science Institute
Department of Computer Science
University of Calgary
Calgary, Alberta,
Canada T2N 1N4

ABSTRACT

Knowledge Support System Zero (KSS0) is a knowledge support system providing an integrated set of tools for knowledge acquisition. Results of using the tools in KSS0 **Elicit, Exchange** and **SOCIO** are reported in the domain of spatial interpolation techniques for contour mapping. A preliminary model for validation of knowledge support systems has been developed and parts of KSS0 have been evaluated against the first stage of the model.

INTRODUCTION

In previous workshops on knowledge acquisition for knowledge-based systems, in it was clear that validation of knowledge acquisition systems is a crucial issue. Shaw and Gaines (1983, 1986) have put great emphasis on the need to be able to validate any technique for knowledge engineering. It is not sufficient to show that reasonable expert systems can be developed; one must attempt to evaluate the accuracy and completeness of the knowledge transfer. This is not an easy task because there are few well-established domains of expertise where the accuracy of the elicitation can be tested. Past studies have been carried out using PLANET (Shaw 1982, Shaw & Gaines 1988) to determine whether PEGASUS can be used to elicit the Business Information Analysis Integration Technique (Carlson 1979, Sowa 1984) distinctions from people with some accounting or business knowledge (Shaw & Gaines 1983). The experiments have been based on taking a spectrum of those with assumed expertise in business record keeping, from businessmen and accountants, to accounting students at various stages of their education, and using PEGASUS to elicit their constructs relating to business transactions.

AN OUTLINE OF KNOWLEDGE SUPPORT SYSTEM ZERO

KSS0 combines a number of different sources, including text, expert interviews and expert decision-making, and a number of different techniques, including text analysis, entity-attribute elicitation, clustering and inductive rule generation. It is written in Pascal and runs on the Apple Macintosh family of computers to provide a highly interactive and graphic knowledge acquisition environment. At the heart of KSS0 is an object-oriented knowledge base in which knowledge is formally represented as a multiple-inheritance digraph of classes, objects and properties. Such a structure generalizes the entity-attribute grids used in several early knowledge acquisition systems (Shaw & Gaines 1983, Boose 1985) and has proved both general and powerful in a variety of applications (Boose & Bradshaw 1986, Gaines 1987).

The main tools in KSS0 are shown in Figure 1:

- **Elicit** accepts specifications of entities within a domain and provides an interactive graphical elicitation environment within which the experts can distinguish entities to derive their attributes. The resultant class is continuously analyzed to provide feedback prompting the expert to enter further entities and attributes.

- **Exchange** allows interactive exchange and comparison of entities and attributes from multiple experts in a domain.
- **FOCUS** hierarchically clusters entities and attributes within a domain prompting the experts to add higher-level entities structuring the domain.
- **PRINCOM** spatially clusters entities and attributes within a domain prompting the experts to add higher-level entities structuring the domain.
- **ENTAIL** induces logical entailments enabling the attributes of an entity, or the evaluations of a decision-making situation in a domain, to be derived from other attributes.
- **SOCIO** compares the structures for the same domain generated by different experts, or the same expert at different times or from varying perspectives.

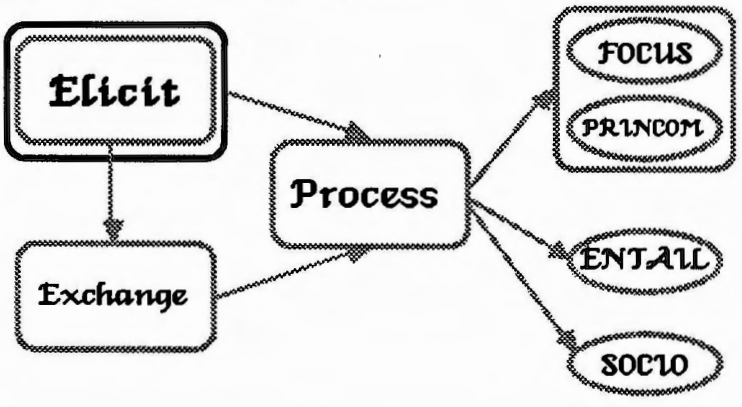


Figure 1. Some tools in Knowledge Support System Zero

Part of the interaction with one of the geographers is used to give a flavour of **Elicit**. Figure 2 shows the common entities agreed on by all the geographersbeing placed on the attribute **local — global.** Some techniques have already been rated, some are waiting in a list on the left, and **proximal mapping** is being dragged on to the attribute bar. Both the attribute labels and the techniques can be edited any time, and also the placing of the techniques. Figure 3 shows two matched attributes, with both sets of ratings shown, so that they can be compared. Again, any one is moveable if the expert wants to adjust anything.

Techniques may also be matched, as shown in the screen in Figure 4. The technique at the top has a marker on each attribute sticking up, and the one at the bottom a marker sticking down. Again, any marker is moveable if the expert wants to change anything. Figure 5 shows the FOCUS clustering of the final grid. As can be seen on the attribute tree at the top right, there are three clusters at 75%, and one attribute which is not part of any cluster. From the bottom right tree of techniques, it can be seen that the one most different from the others is **hand contouring**. Many other conclusions can be drawn from this diagram, but it is sufficient here just to give a flavour of what the expert experienced in using KSS0.

The **Process** facility allows single and multiple grids to be analysed. The **FOCUS** algorithm is a distance-based hierarchical cluster analysis technique that sorts the attributes into a linear order such that those closest together in the space are also closest together in the order. It has the advantage in presentation that the sorting is used only to re-present the original grid re-organized by the neighbourness of attributes and entities. It is left to the user to construe his own personal meaning into the result and confirm this directly in terms of the original data. The tree diagrams show the entity and attribute clusters and are imposed on the re-sorted original grid data (Shaw 1980). The program provides a hierarchical clustering of an expert's conceptual system that preserves the data elicited from him so that the sources of the analysis are evident and can be

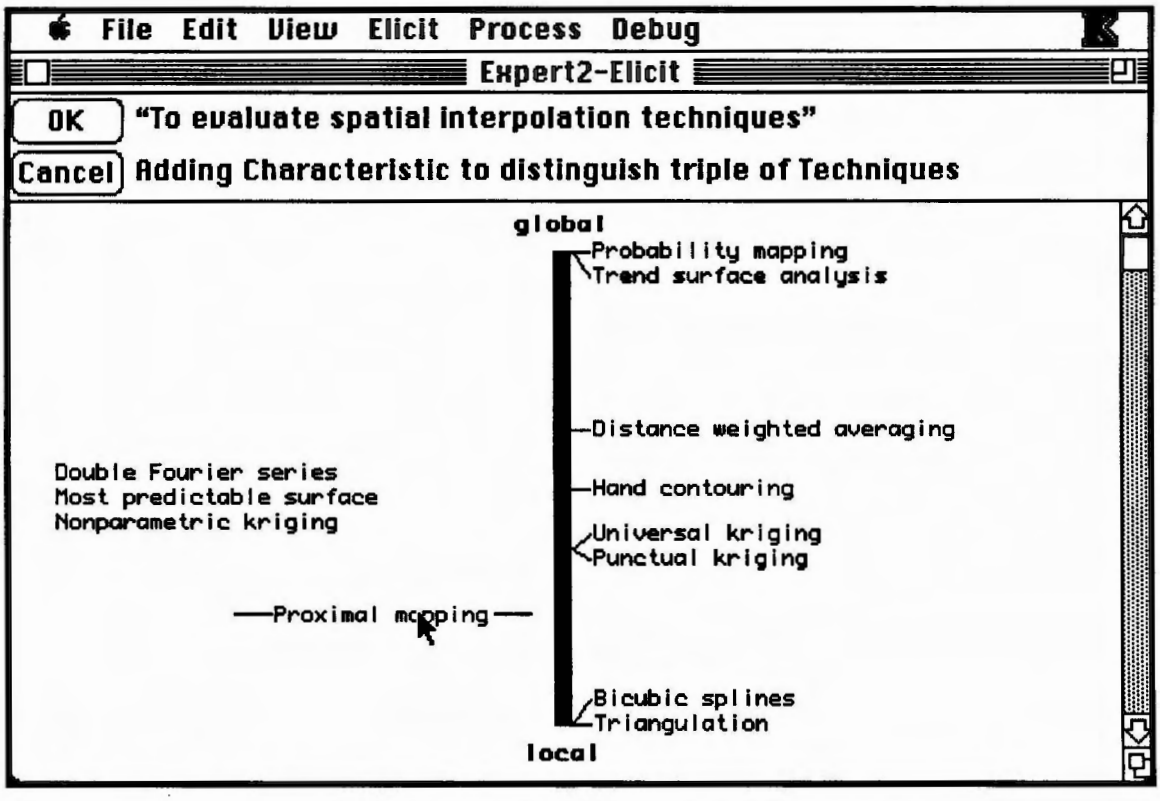


Figure 2. KSS0 showing click and drag elicitation.

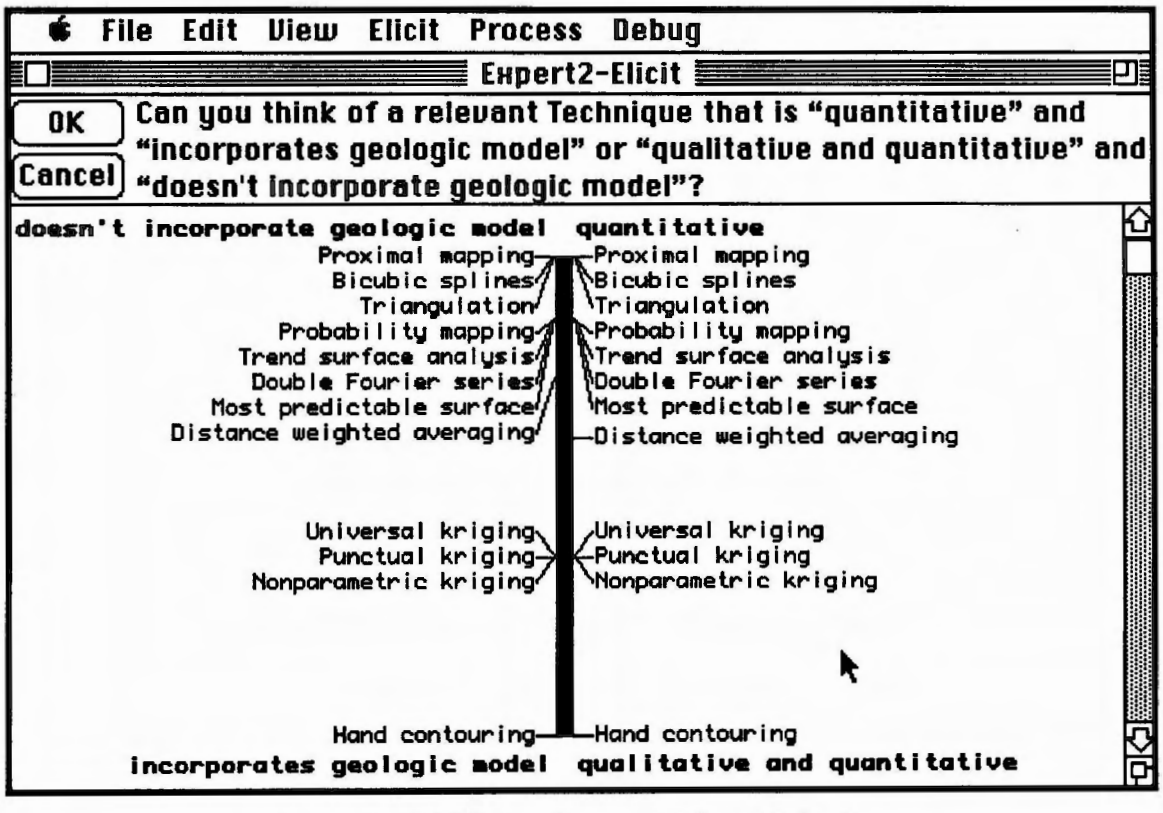


Figure 3. KSS0 showing matching attributes.

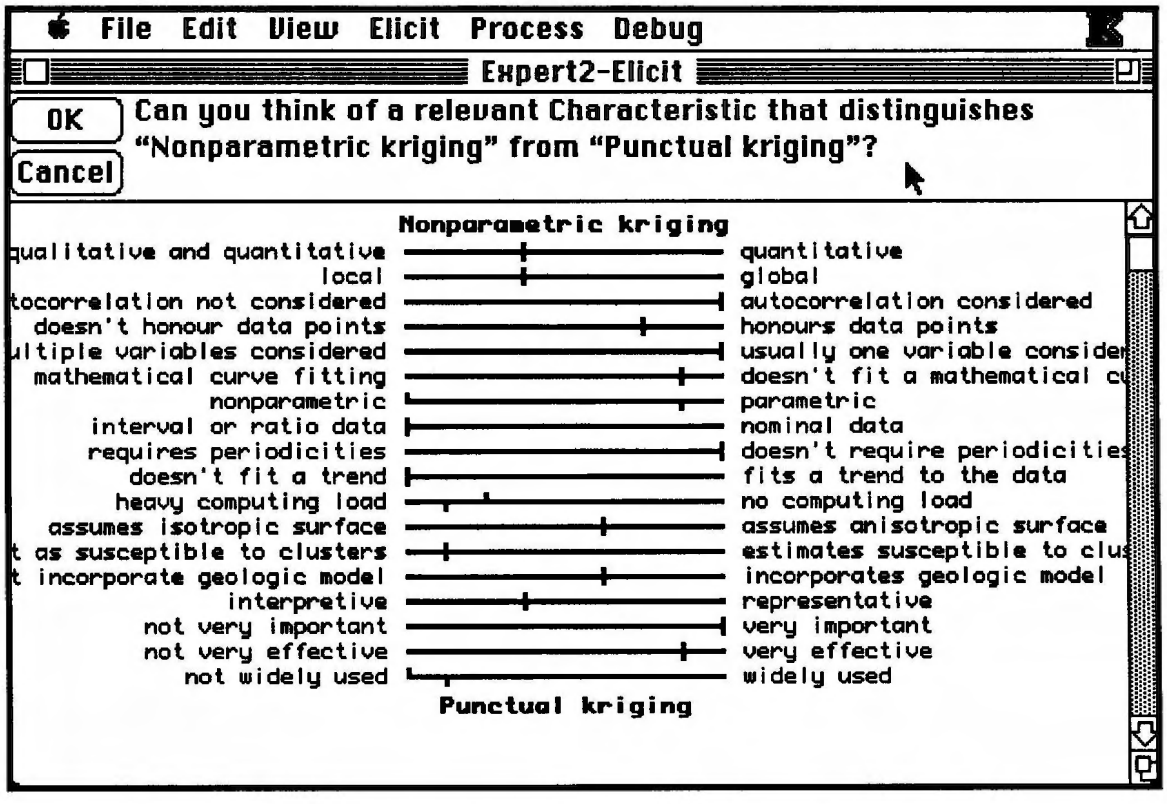


Figure 4. KSS0 showing matching techniques.

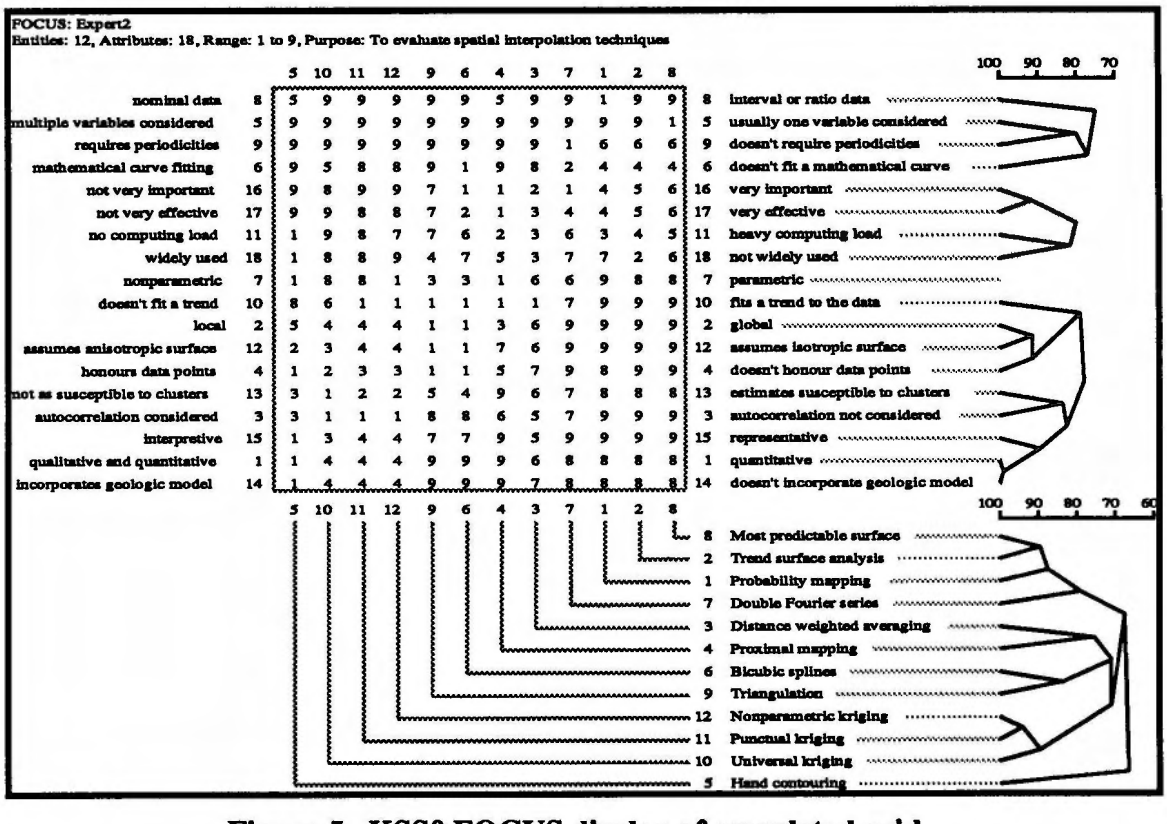


Figure 5. KSS0 FOCUS display of completed grid.

discussed. The **PRINCOM** algorithm is another distance-based cluster analysis using standard principal component analysis techniques and giving the same results as Slater's INGRID (Slater 1976, 1977). This analysis has been used widely because it gives a visually meaningful map of some of the relations between entities and attributes. The program gives a non-hierarchical cluster analysis based on principal components that can be used to gauge the major dimensions along which an expert is making distinctions.

The **ENTAIL** algorithm is a logical analysis of the attribute system taking the expert's distinctions to be fuzzy predicates (Zadeh 1972, Gaines 1976). The program derives asymmetric implications between the attribute poles so that one can infer how a new entity might be placed on one attribute given how it is placed on others (Gaines & Shaw 1981). It can also provide rules for input to an expert system shell (Boose 1985, Gaines & Shaw 1986). **ENTAIL** gives a dependency analysis of the attributes in a grid by deriving significant logical entailments consistent with the data. From a clustering point of view this analysis is significant because it can show up one-way relations that may be missed by distance-based clustering techniques. However, in the present context it is more significant that the entailments may be treated as inference rules to be used in an expert system. Figure 6 shows some results from the **ENTAIL** analyis of the grid of Figure 5. The entailments are shown with two values in the range from 0 to 1: first, the truth value of the hypothesis and second, the information content (uncertainty reduction generated) of asserting the hypothesis. For example, L1 → R14 has a truth value of 0.88, and an information content of 0.25. The information content measures the significance of the hypothesis and is used to ensure that trivial entailments consistent with the data are pruned (Gaines & Shaw 1986).

```
ENTAIL     Truth   Information (Cutoff 0.15) Entailment

R14 -> L 1  1.00  0.30  incorporates geologic model -> qualitative and quantitative
R 1 -> L14  1.00  0.26  quantitative -> doesn't incorporate geologic model
L 1 -> R14  0.88  0.25  qualitative and quantitative -> incorporates geologic model
L15 -> R14  0.75  0.25  interpretive -> incorporates geologic model
L15 -> L 1  0.75  0.24  interpretive -> qualitative and quantitative
R 4 -> R12  0.75  0.22  honours data points -> assumes anisotropic surface
R16 -> R17  0.88  0.22  very important -> very effective
L17 -> L16  0.88  0.22  not very effective -> not very important
L 1 -> L15  0.88  0.21  qualitative and quantitative -> interpretive
L 4 -> L12  0.88  0.21  doesn't honour data points -> assumes isotropic surface
L12 -> L 4  0.75  0.21  assumes isotropic surface -> doesn't honour data points
R14 -> L15  0.88  0.21  incorporates geologic model -> interpretive
R12 -> R 4  0.88  0.20  assumes anisotropic surface -> honours data points
L 4 -> R 2  0.75  0.19  doesn't honour data points -> global
R 3 -> R14  0.62  0.18  autocorrelation considered -> incorporates geologic model
R 3 -> L15  0.62  0.18  autocorrelation considered -> interpretive
L12 -> R13  0.75  0.18  assumes isotropic surface -> estimates susceptible to clusters
L12 -> R15  0.88  0.18  assumes isotropic surface -> representative
L13 -> L15  0.62  0.18  not as susceptible to clusters -> interpretive
L15 -> R12  0.88  0.18  interpretive -> assumes anisotropic surface
```

Figure 6. Some rules produced by the **ENTAIL** algorithm from Expert 2.

Exchange grids are used for the measurement of understanding and agreement between either two experts or on two occasions (Shaw 1980). To do this two people, possibly experts with differing points of view, each elicit a grid in an area of common knowledge or experience. Each may choose his own entities independently of the other, and elicit and rate his attributes quite separately. Each then can **Exchange** his grid, that is use the other's entities and attributes but not the rating values. This is then completed by the other expert. By comparing pairs of these grids it is possible to map the extent of overlap of the agreement and understanding between the two experts. The **Exchange** procedure allows multiple experts to be aware of and explore the meaning of each other's terminology in the domain. The experts can then extend and understand their own thinking

and problem-solving capabilities with ideas from other experts. They are able to see the relationship of their points of view to those of others; explore differing terminology for the same attributes; become aware of differing attributes having the same terminology; extend their own conceptual systems with those of others; provide others with attributes they have found valuable; and explore a problem-solving domain using the full group resources.

The **SOCIO** program provides facilities for comparing and contrasting multiple sources of expertise. It is an extension of SOCIOGRIDS (Shaw 1980) for deriving socionets and mode constructs from groups of individuals construing the same class of entities. Its objective is to take different classes representing the same domain and compare them for their structure, showing the similarities and differences. It may be regarded as the implementation of a simple form of analogical reasoning. The grids may have the same entities and attributes but possibly differing values. **SOCIO** analyzes the matches between the entities and the attributes in the grids according to the values, and shows those entities and attributes that are similar and those which are different. A typical application is to see whether experts agree on the definitions of classes by asking them to separately fill in the values for a domain definition **Exchanged** between them. If the grids have different attributes but the same entities **SOCIO** analyzes the matches between the attributes in the grids, and for each attribute in the original shows the closest matching attribute in the other class. A typical application is to see whether experts are using different terminologies for the same attributes by asking them to separately define attributes and fill in the values for a domain defined through an agreed set of entities.

When a number of grids representing the same domain are available, **SOCIO** also provides two other forms of analysis. The first is that it tries to derive "modal" entities or attributes that reflect a consensus among the experts by extracting those which occur as highly matched entities or attributes across the majority of classes. A typical application is to reach consensus on critical concepts that are associated with a rich vocabulary at differing levels of abstraction. The other is that it derives a set of socionets showing the degree to which each expert is able to make the same distinctions as another expert, even if they use different terminology. A typical application is to validate the knowledge acquisition process by determining whether the structure derived conforms with known relations between the experts. Some output is shown later in Figures 12 and 13.

EVALUATION OF KNOWLEDGE-BASED TOOLS

Over twenty years ago, validation was labeled as the most elusive of all the unresolved problems associated with computer simulation techniques (Naylor, Balintfy, Burdick & Chu 1966) and, despite work which has gone on in the intervening years, it is still an unresolved problem. The main problem with the evaluation of knowledge acquisition systems is that it is almost impossible to separate the use of them from the experts involved. A knowledge-base is produced from this interaction, and it is often the adequacy of this knowledge-base, or the number of rules produced which is measured. This, however, is still more of a test of the expert than the knowledge acquisition system. How significant is the user interface to the knowledge acquisition system? It is very important to the quality of the resulting knowledge-base that the expert is motivated and finds it easy to use the system. These are all questions of **validity** in its general sense of the degree of homomorphism between a representation system and the system that it is supposed to represent (Vandierendonck 1975).

Evaluation studies of knowledge-based systems (specifically expert systems) have been reported and methods for validation have been outlined. For example Siegel (1986) points out the need to evaluate the output of expert systems and outlines in detail the methods for accomplishing this task. He includes criterion variables and the methods for testing. These methods are based on the use of test cases as a developmental tool through successive improvements of the system. Test cases provide standard input to the system and output can be tested for consistency. O'Keefe, Balci and Smith (1987) discuss problems in the validation of expert systems and elaborate the distinction

between validation and verification: **validation** means "building the right system" and **verification** means "building the system right". At the previous European knowledge acquisition workshop Burton, Shadbolt, Hedgecock and Rugg (1987) reported an evaluation study of a number of knowledge acquisition techniques: interview, protocol analysis, goal decomposition, and card sort in the domain of rock identification.

Although many knowledge acquisition systems have been designed, relatively little has been deduced about their evaluation. The following section outlines a framework for the evaluation and validation of knowledge acquisition systems (Shaw & Woodward 1988).

FRAMEWORK FOR EVALUATION

In experimental psychology some "dependent" variable acts as the starting point and various factors are hypothesized which might predict that variable. In evaluation research, the independent variable (e.g. the program, the system, the procedure) acts as the starting point and assessments are made against a set of criteria or goals (Bernstein, 1976). Evaluation research is expected to tell how well something "works": how well something does what it is supposed or conceived to do. Evaluation can also be structured to tell when, why and how something works.

In order to be effective, evaluation designs require clear explication of components and requirements: determine the external criteria which act as comparison points for the system under evaluation; recognize levels of operation and measurement in the external criteria; determine and describe the input and output of the system under evaluation; require clarification, justification and theoretical support of the methods of processing inputs to outputs (e.g. the algorithms); carefully select populations for study.

This section outlines an evaluation framework for knowledge support systems. Applying the above criteria to knowledge support systems, evaluation stages and specific evaluation tasks are outlined for the evaluation activity. General design considerations have been elucidated for knowledge support systems (Shaw and Gaines, 1987). These are also incorporated into the evaluation framework.

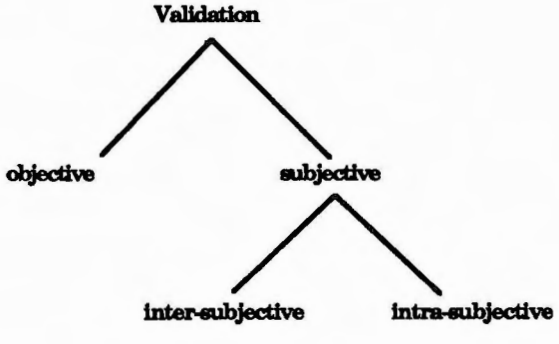


Figure 7. A validation model.

Figure 7 shows two distinct aspects of validation: those concerned with objective aspects of the system and those concerned with subjective aspects of the system. These distinctions represent the various aspects of operation of knowledge support systems. The objective aspects are those concerning the performance of the expert system or knowledge base, elicited from the expert or experts, against objective criteria. A knowledge support system supports the knowledge processes of the expert, and allows her to develop a system which actually performs effectively in the real world, rather than merely confirming the opinion of the expert. The study done by O'Keefe et al looks at objective aspects of a system.

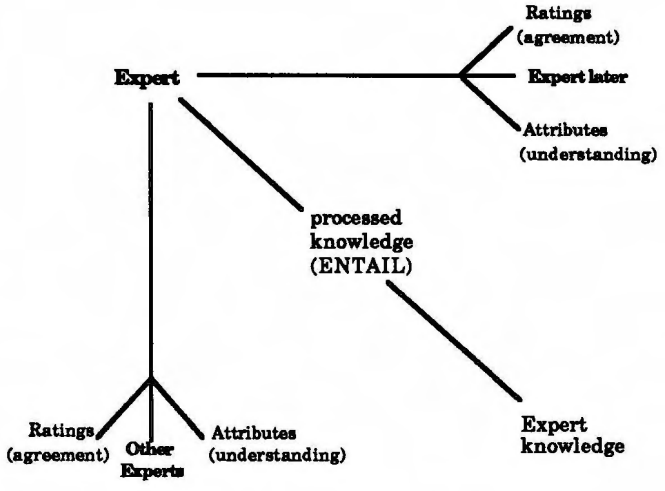


Figure 8. The framework for the study.

The subjective aspects relate to the way in which the knowledge acquisition system acquires the knowledge that the expert has. Since it is impossible to separate the knowledge acquisition tool from the expert, so the term **knowledge acquisition system** will be used to mean the knowledge acquisition tool, **and** the expert:

knowledge acquisition system = knowledge acquisition tool + expert

Figure 8 shows the model developed for this study. It is preferable to have multiple experts involved in building a knowledge base, so it was necessary to see if there was mutual agreement across experts. Also, whether there was any consistency of an expert over a time interval, both in the terminology used, and how it was used. Finally, an investigation was made as to whether the **ENTAIL** analysis performed by the system on each expert's knowledge was seen as correct, or if any of it was incorrect.

METHODOLOGY

Figure 9 outlines the procedural steps and the data collection points used in the subjective evaluation of KSS0 (Shaw & Woodward 1988). It shows the first step as Procedure 1 (P1) which introduced the subjects to the requirements of KSS0. At this time a specific task was agreed upon and a purpose for the grid elicitation was developed. The first data collection point (G1) consisted of the elicitation of a grid by each subject individually. This provided an initial set of entities and attributes particular to each subject. The second collection point (G2) consisted of the subjects exchanging their first grids (entities and attributes) to produce other grids. The next step was a procedure (P2) to have the subjects agree on a common set of entities. This common entity set provided the basis for the next data collection point (G3) which consisted of the elicitation of a grid by each expert using common entities. This elicitation was done, as was the other, separately for each expert. P3 represents a wait of over two months before the next data collection point (G4). At this time a repeat of the activity at G3 was completed using a common entity set but the regeneration of attributes. For G5 the subjects were given the entity and attribute set from G3 and asked to re-rate the entities. The final data collection point (P4) consisted of giving the subjects a selected list of entailments using the **ENTAIL** algorithm (Gaines & Shaw 1986) generated from the G3 grid. A set of examples from four separate levels of significance of entailment were selected and randomly presented to the subjects who were asked to rate each entailment on a four point scale from correct to incorrect.

Event	Description
P1	determine task demands and elicitation purpose
G1	grid elicitation using own entity set
G2	grid eliciation using other expert's entities and attributes
P2	experts discuss and produce a common entity set
G3	grid elicitation using common entity set
P3	waiting period: 2 to 3 months
G4	grid elicitation using G3 entity set
G5	grid elicitation using entities and attributes from G3
P4	obtain subjective ratings from entailment output from G3
	Analysis

Figure 9. Evaluation design event table.

Two separate studies were completed using specific task demands from two separate and quite different domains, but only the geography one will be reported here. It involved three researchers in a university geography department each of whom is an expert working in the field of spatial interpolation techniques to produce contour maps. Each was interviewed separately (P1) and was involved independently for an hour or more in the first session on the task of "evaluating spatial mapping techniques" (G1). In the second session, on the following day, each expert exchanged his or her attributes with the others. This involved Expert 1 in seeing Expert 2's entities and attributes without the ratings, and rating them as he would have done (G2a). He repeated this for Expert 3's grid (G2b). This was done for each expert in turn.

Following this session, all three experts met together to discuss the entities each had used, and to agree both on the meanings of each, and on a common set that all could use. There were twelve of these decided after 30-40 minutes. In the third session, in the same week, each expert elicited another set of attributes which could include the previous ones elicited, as well as any he or she had seen when doing the exchange procedure. This was G3. Three attributes were added at this stage by the experimenter: **very important -- not very important; very effective -- not very effective;** and **widely used -- not widely used.** After this session, the experts were asked not to think about the topic more than they usually would, not to discuss the topic with each other, and not to review their attributes before the next session.

In the fourth session, twelve weeks later, each expert in turn elicited another set of attributes using the agreed set of entities from session three producing G4. Immediately this was completed, in about an hour, each expert was shown his entities and attributes from session three, that is, G3 with the ratings removed, and asked to re-rate these on the same scale. This gave G5. The entailments (Gaines & Shaw 1986, Shaw & Gaines 1987) were produced for each expert from G3. These rules were then sampled from each category of significance, $\geq .15$, .14 to .10, and .09 to .05 and randomly presented to the expert (P4) who was asked to rate them on a scale 1- correct, 2 - likely to be correct, 3 - not likely to be correct, and 4 - incorrect.

In the first session, two of the experts used a 5-point scale, and one a 9-point. When agreeing on their common set of entities, they also agreed to use a 9-point scale. Although this seems cumbersome when using numbers it is, in fact, a finer discrimination of a 5-point scale, having all the in-between points available for finer distinctions on the spatial attribute bar. All three experts agreed that the system was easy to use, allowed easy correction of errors, and gave them the options that they needed to peruse the data.

THE MEASURES

Two measures were used in this study, **consistency-with** another and **construed-by** another. **Consistency** is measured using **Exchange**. Do the experts see the topic in a similar way at the same time? By exchanging entities and attributes they are able to view the topic from the perspective of the other. To do this involves an understanding of what the other's entities and attributes actually **mean** in the domain, or **can I construct a point of view which makes sense of them?** Not only that, but **do I agree with the other's construction of the topic, having understood the perspective of the other?** So **consistency** is defined as the degree of match between one expert's ratings of his/her own entities and attributes and another expert's ratings of the first expert's entities and attributes. The patterning of the entities on a particular attribute is matched against the patterning of the same entities on that same attribute in the other grid.

Consistency over time can also be measured in this way but using the **same** expert's grid on the two occasions with the same entity and attribute labels as before, but new ratings. Only entities and attributes with the same labels in both grids can contribute to this measure of consistency. Operationally, this measure was calculated using the **SOCIO** measure based on the matching algorithm that is described in Shaw (1980) and used in FOCUS, CORE and SOCIOGRIDS (Shaw 1980). It ranges from 100 for perfect match to 0 for maximum reversal. This measure is picking out the differing use of terminology in the two grids. The following formula was used:

$$\textbf{G consistency-with G' at criterion} = 100 \times \frac{\text{(number of attributes in G matched greater than criterion by same attributes in G')}}{\text{(number of attributes in G)}}$$

Construing is akin to consistency. The entity labels must be the same in both grids, but separate sets of attributes can be used. In this case, the measure picks out the attribute in the second grid with the best match to the patterning of the entities on the one being considered in the first grid. In this case, the same attribute may be chosen more than once in the second grid if it matches more closely than any other to more than one attribute in the first. Again, the same matching algorithm was used. This measure is picking out the best match between the attributes in the two grids, regardless of terminology. The following formula was used:

$$\textbf{G construed-by G' at criterion} = 100 \times \frac{\text{(number of attributes in G matched greater than criterion by any attributes in G')}}{\text{(number of attributes in G)}}$$

Operational definitions for the various replication and consistency measures were used to evaluate the knowledge support system. **Intra-subjective understanding, U_t,** was defined as the degree of construing of the earlier grid G3 by the later one G4. Those attributes which were matched on the G3 and G4 grids at a criterion of 80 were used for this calculation, and this was calculated for each expert:

U_t = G3 construed-by G4 at 80

Intra-subjective agreement, A_t, was defined as the degree of consistency between one expert's ratings of his/her own entities and attributes in G3 with his/her ratings of the entities and attributes in the later grid G5. The following formula was used to calculate A_t for each expert:

A_t = G3 consistency-with G5 at 80

Inter-subjective understanding, U_i, was defined as the degree of construing of each expert's grid G3 by another expert's G3:

U_i = G3E construed-by G3E' at 80

Inter-subjective agreement, A_i, was defined as the degree of consistency between one expert's ratings of his/her own entities and attributes in G1 with another expert's ratings of the first expert's entities and attributes in G2. The following formula was used:

A_i = G1E consistency-with G2E' at 80

[N.B. There were two G2s: G2a and G2b, so two measures resulted for each expert.]

Finally, **intra-subjective perspective consistency, C_i,** was defined as the degree of consistency of the rules produced by the **ENTAIL** algorithm with what the expert apparently expected. The ratings of these statements at P4 were then compared to the entailment output from G3. The number of identical ratings and the number of similar ratings (differing by only one value) were selected. The following formula was used to determine C_i:

$$C_i = \frac{\text{(number of same ratings + number of similar ratings)}}{\text{(total number of rules given)}} \times 100$$

RESULTS

Study 1	$U_t \geq 80$	$A_t \geq 80$
E1	62.5	81.2
E2	77.8	94.4
E3	85.7	78.6
Study 2		
E1	100	66.7
E2	83.3	66.7

Figure 10. Intra-subjective agreement (A_t) and understanding (U_t) results.

Expert Pairs	Ui ≥80	Ai≥ 80
Study 1		
E1, E2	62.5	33.3
E2, E1	61.1	26.7
E1, E3	31.2	8.3
E3, E1	42.9	33.3
E2, E3	44.4	20.0
E3, E2	71.4	33.3
Study 2		
E1, E2	100	12.5
E2, E1	66.7	100

Figure 11. Inter-subjective agreement (A_i) and understanding (U_i) results.

Figures 10 to 14 display the results. The **intra-subjective understanding** scores, U_t, displayed in Figure 10 ranged from 63 to 86; and the **intra-subjective agreement** scores, A_t, from 79 to 94, indicating that each expert was using her/his terminology in much the same way from one occasion to the other and that KSS0 was able to reflect this consistency.

```
Grid  Ent Att
* E1:  11  12  Expert1
  E2:  11  12  Expert2
  E3:  11  12  Expert3
       11  12  Common Entities & Common Attributes

E1:E2 33.3% over 80.0 (Expert1 attribute-consistency-with Expert2)

 1:   8.3% > 90.9  A2: Interval data - Nominal data
 2:  16.7% > 81.8  A4: Global - Local
 3:  25.0% > 81.8  A5: Intuitive - Mathematical
 4:  33.3% > 81.8  A6: Requires spatial search - Does not require spatial search
 5:  41.7% > 75.0 A10: Difficult to understand - Easily understood
 6:  50.0% > 72.7  A3: Non-polynomial - Polynomial
 7:  58.3% > 72.7  A7: Discontinuous - Continuous
 8:  66.7% > 59.1 A12: Does not consider non-spatial attributes - Considers non-spatial attributes
 9:  75.0% > 56.8 A11: Few points - Many points
10:  83.3% > 50.0  A8: Does not honour data - Honours data
11:  91.7% > 47.7  A1: Requires no model - Requires model
12: 100.0% > 47.7  A9: Linear interpolation - Non-linear interpolation

E1:E3 8.3% over 80.0 (Expert1 attribute-consistency-with Expert3)

 1:   8.3% > 86.4  A2: Interval data - Nominal data
 2:  16.7% > 79.5  A5: Intuitive - Mathematical
 3:  25.0% > 79.5  A7: Discontinuous - Continuous
 4:  33.3% > 77.3  A4: Global - Local
 5:  41.7% > 75.0  A3: Non-polynomial - Polynomial
 6:  50.0% > 68.2 A11: Few points - Many points
 7:  58.3% > 65.9  A1: Requires no model - Requires model
 8:  66.7% > 63.6  A9: Linear interpolation - Non-linear interpolation
 9:  75.0% > 63.6 A10: Difficult to understand - Easily understood
10:  83.3% > 56.8  A8: Does not honour data - Honours data
11:  91.7% > 56.8 A12: Does not consider non-spatial attributes - Considers non-spatial attributes
12: 100.0% > 52.3   A6: Requires spatial search - Does not require spatial search
```

Figure 12. Analysis of inter-subjective agreement.

Inter-subjective agreement scores, A_i, are displayed in Figure 11. That is the extent to which the experts agree, range from 8 to 33. This shows that the experts disagree with each other in their terminology and in how they view the topic quite extensively. This can be seen in Figure 12 where the details of the **SOCIO** analysis are shown. E1 has exchanged his techniques and attributes with the other two, E2 and E3. The listing at the top shows the agreement for the pair E1, E2. The highest matched attribute is **interval data — nominal data**, and four attributes are agreed upon over the threshold of 80. In the other listing for E1, E3 the highest matched attribute was also **interval data — nominal data** but was the only one matched above the threshold of 80. Inspection of the raw grids for E1 and E2 show the reason for these results. The first number on the first line is a 1 in E1's grid and a 5 in E2's. This means that E1 thinks that **probability mapping requires model**, whereas E2 thinks that **probability mapping requires NO model**. On inquiring about this, the explanation given was couched in terms of what one actually means by the term **model**, indicating the difference in terminology which was shown up in the measures of A_i.

Inter-subjective understanding scores, U_i, are also shown in Figure 11. Here the values are much higher showing that the experts have similar distinctions about the topic even though they differ greatly in how they use the terminology. These values range from 31 to 71, and Figure 13 shows the matches which were found for the pair E1, E2 — the first attribute coming from E1 and the second being the one from E2 which matched the highest **patterning** of the techniques on the first attribute. This was repeated for each of E1's attributes.

```
Grid  Ent Att
* E1:  12  16  G3
  E2:  12  18  G3
       12  34  Common Entities & Total Attributes

E1<:E2 62.5% over 80.0 (E1G3 attribute-construed-by E2G3)

 1:   6.2% > 88.5    E1A2: Local - Global
                     E2A2: local - global
 2:  12.5% > 87.5    E1A3: Low level data - High level data
                     E2A8:   nominal data - interval or ratio data
 3:  18.8% > 86.5    E1A1: Does not honour data points - Honours data points
                     E2A4:  doesn't honour data points - honours data points
 4:  25.0% > 86.5    E1A7: Short distance autocorrelation - Long distance autocorrelation
                     E2A2:                              local - global
 5:  31.2% > 86.5    E1A9: New geographical technique - Old geographical technique
                    E2A18:             not widely used - widely used
 6:  37.5% > 85.4   E1A16: Not widely used - Widely used
                    E2A18: not widely used - widely used
 7:  43.8% > 83.3    E1A5: Discontinuous - Continuous
                     E2A2:          local - global
 8:  50.0% > 82.3    E1A4: Mathematically complex - Mathematically simple
                    E2A11:   heavy computing load - no computing load
 9:  56.2% > 81.2   E1A10:   Hard to adapt to multivariate - Easy to adapt to multivariate
                     E2A5: usually one variable considered - multiple variables considered
10:  62.5% > 81.2   E1A12:   Does not require spatial search - Requires spatial search
                    E2A13: estimates susceptible to clusters - not as susceptible to clusters
11:  68.8% > 79.2    E1A6: Does not require a priori model - Requires a priori model
                     E2A2:                                  local - global
12:  75.0% > 79.2   E1A11:      Few points - Many points
                    E2A18: not widely used - widely used
13:  81.2% > 76.0   E1A13:          Does not use polynomial - Uses polynomial
                     E2A6: doesn't fit a mathematical curve - mathematical curve fitting
14:  87.5% > 76.0   E1A15: Not very effective - Very effective
                    E2A17: not very effective - very effective
15:  93.8% > 72.9   E1A14: Not very important - Very important
                    E2A18:    not widely used - widely used
16: 100.0% > 71.9    E1A8:        Models the stationarity - Assumes stationarity
                     E2A3: autocorrelation not considered - autocorrelation considered
```

Figure 13. Analysis of inter-subjective understanding.

These results suggest that KSS0 provides a facility for allowing experts to compare their understanding and to determine a level of consistency. Whether the threshold criteria are high enough to produce the required level of knowledge-base performance or too high to reach the criterion of completeness is a matter for further study.

Intra-subjective perspective consistency scores C_i are displayed in Figure 14. These show whether the expert finds the rules meaningful, and rates the rules as correct or significant in the same way as the **ENTAIL** (Gaines & Shaw 1986, Shaw & Gaines 1987) algorithm. They range from 82 to 92, showing a high level of expected rules appearing in the KSS0 output. This indicates a good basis for using the **ENTAIL** produced rules as input to an expert system shell. A few of these rules from expert E2 were shown in Figure 6.

Study 1	CI
E1	88.2
E2	91.9
E3	82.2

Figure 14. Intra-subjective perspective consistency results (C_i).

CONCLUSIONS

KSS0 is a knowledge support system providing an integrated set of tools for knowledge acquisition. **Elicit** provides facilities for eliciting the important dimensions of an expert's thinking on a topic; **Exchange** extends this to share entities and attributes between experts and elicit differences in perspective and terminology as well as disagreements on the topic. **SOCIO** processes results from several experts to reveal the similarities and differences in the concept systems of different experts, or the same experts at different times, construing a domain defined through common entities or attributes. It can be used to focus discussion between experts on those differences between them which require resolution, enabling them to classify them in terms of differing terminologies, levels of abstraction, disagreements, and so on.

Some preliminary studies have been reported on validating this knowledge support system, and measures have been put forward which tease out differences in terminology between experts, and show the content of this difference. To a knowledge engineer, it seems that if experts talk about their topic of expertise in the same terms then they mean the same thing, and if they talk in different terms they mean different things. However, it can be seen from the results of this study that the knowledge engineer, user of the system, or even another expert, may be mistaken. The experts were clearly accustomed to disagreeing on terminology, a fact which may not always be made explicit. The consistency measure used for inter-subjective agreement showed up the divergence in use of terminology between the experts, but when used for one expert alone for intra-subjective agreement gave much higher values. This measure, then, shows a good range of discrimination within experts and between experts. A similar rationale can be applied in the construed-by measures of inter- and intra-subjective understanding.

ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. The KSS0 system is made available by the Centre for Person Computer Studies. I would like to thank my colleagues Brian Gaines and Brian Woodward for discussion of this work.

REFERENCES

Bernstein, I.N. (1976) **Validity Issues in Evaluation Research**. Beverly Hills: Sage.

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. & Bradshaw, J.M. (1986) Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop**. pp.5-0-5-16. Banff (November).

Burton, A.M., Shadbolt, N.R., Hedgecock, A.P. & Rugg, G. (1987).A formal evaluation of knowledge elicitation techniques for expert systems. **Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge-Based Systems**. Reading, U.K.

Carlson, W.M. (1979) The new horizon in business information analysis. **Data Base, 10**(4), 3-9.

Gaines, B.R. (1976). Foundations of fuzzy reasoning. **International Journal of Man-Machine Studies, 8**(6), 623-668 (November).

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

Gaines, B.R. & Shaw, M.L.G. (1981). New directions in the analysis and interactive elicitation of personal construct systems. Shaw, M.L.G., Ed. **Recent Advances in Personal Construct Technology**. pp. 147-182. London: Academic Press.

Gaines, B.R. & Shaw, M.L.G. (1986) Induction of inference rules for expert systems. **Fuzzy Sets and Systems 18**(3) 315-328 (April).

O'Keefe, R.M., Balci, O., & Smith, E.P. (1987) Validating expert system performance. **IEEE Expert 2**(4) 81-90 (Winter).

Naylor, T.H., Balintfy, J.L., Burdick, D.S. & Chu, K. (1966). **Computer Simulation Techniques**. New York: John Wiley.

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

Shaw, M.L.G. (1982) PLANET: some experience in creating an integrated system for repertory grid applications on a microcomputer. **International Journal of Man-Machine Studies, 17**(3), 345-360.

Shaw, M.L.G. & Gaines, B.R. (1983) A computer aid to knowledge engineering. **Proceedings of British Computer Society Conference on Expert Systems**, 263-271 (December). Cambridge.

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) Techniques for Knowledge Acquisition and Transfer. **International Journal of Man-Machine Studies, 26(3),** (in press).

Shaw, M.L.G. & Gaines, B. R. (1988). PLANET: a computer-based system for personal learning, analysis, negotiation, and elicitation techniques. Mancuso, J.C. & Shaw, M.L.G., Eds. **Cognition and Personal Structure** . pp. 27-65. New York: Praeger.

Shaw, M.L.G. & Woodward, J.B. (1988) Validation in a knowledge support system: consistency and.construing with multiple experts. **International Journal of Man-Machine Studies,** in press.

Siegel, P. (1986) **Expert Systems: A Non-Programmer's Guide to Development and Application.** TAB, Blue Ridge Summit, PA.

Slater, P., Ed. (1976). **Dimensions of Intrapersonal Space: Volume 1.** London: John Wiley.

Slater, P., Ed. (1977). **Dimensions of Intrapersonal Space: Volume 2.** London: John Wiley.

Sowa, J.F. (1984) **Conceptual Structures: Information Processing in Mind and Machine.** Reading, Massachusetts: Addison-Wesley.

Vandierendonck, A. (1975) Inferential simulation: hypothesis-testing by computer simulation. **Nederlands Tijdschrift voor de Psychologie, (Amsterdam) 30,** 677-700.

Zadeh, L.A. (1972). Fuzzy languages and their relation to human intelligence. **Proceedings International Conference on Man and Computer.** Basel: Karger.

From Information Analysis Towards Knowledge Analysis

Steven Twine

Department of Computer Science,
University of Queensland,
St Lucia, 4067.
AUSTRALIA.

ABSTRACT.

There is a lot of interest in exploring the relationships between knowledge bases (in Artificial Intelligence) and databases (in Information Systems). Most researchers view this relationship at the implementation level, and concentrate on the transfer of data structures and processing techniques from one research world to the other. Some researchers have also noted a strong resemblance between the task of knowledge acquisition and the task of information analysis (or conceptual schema design).

A more fundamental view of this relationship is concerned only with the logical structure of the information contained in the database or knowledge base. At this *conceptual* level, both databases and knowledge bases are simply collections of facts (sentences) about the Universe of Discourse. The conceptual level completely excludes any implementation or user presentation aspects, but includes linguistic aspects (which can be effectively used for design and validation purposes).

Using the conceptual level as a middle ground, we can partition the data/knowledge base design process into three parts: acquisition of the 'raw material', analysis and formalization of this information at the conceptual level (possibly requiring the acquisition of additional information), and finally, the implementation of this formal description in the data structures of some program or database management system.

Typically, the knowledge engineer has little effective guidance in identifying, formalizing and representing the relevant elements of the Universe of Discourse. Too often, this forces the knowledge engineer to adopt an intuitive, informal approach to the problem, which emphasizes the construction of an executable prototype rather than the formal analysis of the Universe of Discourse. This means that the expert must validate the system at the (unfamiliar) *internal* level, and not at the conceptual level (where familiar linguistic constructs are available).

One of the most urgent needs is for a methodology (or design procedure) which offers the knowledge engineer very precise guidance on how to perform this complex task.

Over the past fifteen years, there has been a good deal of research directed towards developing well-guided methodologies for information analysis. Some of these information analysis methodologies can be immediately applied to the *strongly related* task of knowledge analysis. This should reduce the complications involved in building, analysing and validating knowledge-based systems, making these difficult tasks significantly easier for the knowledge engineer by providing prescriptive guidance. In this article, I describe the NIAM information analysis methodology and discuss how it can be immediately applied to the tasks of knowledge acquisition, analysis and representation.

1. INTRODUCTION.

Classically, knowledge acquisition for expert systems is viewed as an *indivisible* process: a grand leap from a body of unsuitably structured information (typically, verbal data) about the Universe of Discourse (UoD) to end up with a formal, computer-oriented representation of that information (as an implementation data structure). The knowledge engineer usually has very limited guidance to help him make this grand leap.

This view of knowledge acquisition presumes that each unstructured element of verbal data represents a complete, and self-contained, proposition about the UoD. This is usually *not* the case. Breuker and Wielinga [2] have observed that "these verbal data do not speak for themselves." Rather, various elements in this informal data must be *interpreted* as formal statements about the UoD under consideration.

This has lead to a new view of knowledge acquisition as a two-step interpretation process.

First, the unstructured data are interpreted, using an *interpretation framework*, to yield a formal representation of the UoD (sometimes called the conceptual model). Then, this formal representation is translated into implementation data structures. In this way, the grand, but unmanageable, leap is reduced to two smaller, and hopefully more manageable, leaps.

Even then, without strong guidance for the knowledge engineer, the initial information will probably be misinterpreted or underinterpreted. This is a common problem. Freiling and Alexander *et al* [1] have observed that most development problems are the result of confusion about the appropriate classification of the various knowledge elements in the UoD. This usually leads to an *ad hoc* choice of abstractions for representing this information, with the result that the conceptual model will be unsuitable for its intended task.

In the knowledge engineering world today, there can be little doubt that one of the most urgent needs is for a *practical* and *prescriptive* methodology for interpreting the unstructured data and then using this interpretation as a basis for choosing appropriate abstractions for the formal representation of the expert's knowledge.

Any proposed methodology for this task must address *three key questions:*

(1) How can the unstructured data be interpreted to build a conceptual model?

(2) How can this conceptual model be effectively validated by the expert?

(3) How do we translate the formal conceptual model into implementation data structures?

The success of a knowledge analysis methodology depends on the answers that it provides for these questions.

Several solutions to this problem have been formulated by the Artificial Intelligence research community. They range from general advice (usually based on the author's personal experiences in building a specific expert system) to detailed, descriptive methodologies (such as KADS [3] or Ontological Analysis [1]). Almost all of these approaches make the *implicit* assumption that knowledge engineering requires design techniques which are fundamentally different from the design technique for other software systems. It is true that knowledge engineering requires a very different methodology to software engineering.

However, knowledge-based expert systems can be considered a (powerful) class of fact-based information systems [14] (assuming a broad definition of the term). If the user enters all the facts explicitly, then the system is called a database system; if the user enters some facts explicitly, and others are deduced by applying *explicit inference rules*, then it can be called an expert system.

This statement provides one basis for understanding and evaluating the extensive overlap between the activities of knowledge acquisition (in the artificial intelligence research world) and conceptual schema design (in the database research world) that has been noticed by several researchers [10], [14], [17].

Taking this relationship between information systems and expert systems as an axiom, we can conclude that the design techniques for knowledge bases (in expert systems) should bear a similarly close relationship to the design techniques for databases at the conceptual level (in information systems).

Information analysis [15], takes a coarse, and apparently informal, specification of a UoD and produces a formal definition of the information to be recorded about that UoD (called a conceptual schema). This is essentially the same task as *knowledge analysis*, which begins with informal, unstructured data and produces a formal conceptual model for the UoD.

The conceptual schema is solely concerned with the semantic structure of this information, and specifically excludes any implementation or user presentation aspects. It acts as a prescriptive grammar to specify which facts (sentences) may be recorded (in the database) about the UoD [19]. A conceptual schema, together with a conforming database, is essentially a knowledge base.

Among the most useful results of the last fifteen years of database research, is a principled information analysis (or conceptual schema design) methodology called NIAM [15], [16]. In this article, I will describe how NIAM can be used for knowledge analysis.

Section 2 describes the essential steps of the NIAM conceptual schema design procedure (CSDP). In particular, I will concentrate on how the expert provides (and validates) an interpretation for the initial data, rather than the knowledge engineer. I will also explain how the entire interaction between expert and knowledge engineer can be conducted using only examples of the expert's work, expressed in the expert's natural language. Finally, I will compare this use of the NIAM CDSP with some other knowledge analysis techniques.

Section 3 presents a deterministic procedure for translating a NIAM conceptual schema into a semantically-equivalent set of KEE frames.

Finally, in section 4, I conclude that a principled information analysis methodology, such as NIAM, can be effectively used for the tasks of knowledge acquisition, analysis and representation for expert systems.

2. KNOWLEDGE ANALYSIS: COLLECTING and INTERPRETING EXAMPLES.

It is helpful to view knowledge acquisition as an act of communication between the expert and the knowledge engineer. The presence of the Feigenbaum Bottleneck [11] clearly indicates that this communication is largely ineffective.

Nijssen [13] has defined two prerequisites for effective communication (later named the HELSINKI PRINCIPLE by an ISO Working Group [19]):

> Before two partners may effectively communicate, they must share a common language in which they can talk, and a common world model so that they attribute the same meaning to the same sentences.

This provides a theoretical explanation for some common knowledge acquisition problems.

In many cases, the expert does not understand the requirements of the knowledge representation language and is unable to formulate her knowledge in the required way. This is called *representation mismatch*; it occurs when the expert and the knowledge engineer do not communicate using the same language. Because the task is to build a knowledge-based system (not

to train a new knowledge engineer), the expert's natural language should be the language of communication. The natural language for any UoD is the language that two experts in the field adopt when communicating with one another.

In other cases, the knowledge engineer misinterprets, or underinterprets, the information provided by the expert. This is called the *interpretation problem*; it occurs when the expert and knowledge engineer are using the same language, but not the same world model. This means that they have different semantic interpretations for the same sentence (in their common language).

In the AI world, the classical solution to the interpretation problem was to translate the expert's information into some implementation data structure (often called a representation language!). The idea was that these data structures (as embodied in a prototype expert system) could serve as an interpretation framework for subsequent acquisition and analysis. Of course, this is no solution at all because these data structures are exclusively computer-oriented rather than user-oriented. Quite simply, they are at the wrong level of description for the expert use them effectively.

In knowledge analysis, the conceptual model is a formal definition of the language (descriptive terms) and world model (semantic constraints) that the expert and knowledge engineer *agree to share*, for the duration of the design process.

This raises the immediate question: how does the knowledge engineer construct this conceptual model?

The knowledge engineer must find the *expert's* interpretation of the 'raw' data, then use this interpretation to choose appropriate *abstractions* for the conceptual model.

Our view of the knowledge acquisition process as a communication process suggests that we take statements in the expert's natural language as the basic data for interpretation. Of course, almost all other knowledge analysis approaches have the same starting point.

In [15] Nijssen observes that most people can cook a good meal if provided with a good cookbook. Reading "knowledge analysis methodology" for cookbook, we can translate this to: most knowledge engineers can design a high-quality conceptual model, if they are given a principled methodology.

This is almost true: they also need good ingredients. Even a *cordon bleu* chef cannot make an good *souffle* with old eggs! So, the NIAM cookbook uses the best available source of raw ingredients: *specific examples, expressed in the expert's natural language.*

This exploits the fact that the expert is extremely familiar with specific examples of the problems that she solves in the course of her everyday work. This is a basic assumption of NIAM design methodology. Moreover, she finds it easy to provide such examples for analysis by the knowledge engineer.

Further, it is important that the set of examples be significant, in that they cover all pertinent aspects of the UoD that is to be formalized. This is very difficult to ensure, so rather than taking this as a basic assumption, the NIAM knowledge engineer formulates additional examples, as directed by various steps in the design procedure, and asks the expert whether these examples are valid or not. In this way, *all* communication between the expert and the knowledge engineer is *at the level of specific examples*, expressed in the expert's normal technical vocabulary.

The problem with these examples is that they may contain ambiguity or other conceptually irrelevant material. The examples by themselves are not sufficient; we also need the *expert's interpretation* of the example data.

For example, consider this example from a medical UoD:

Patient	Temp.	Symptoms	Diagnosis
Tommy	40.8	Red Spots, Itchy	Measles
George	37.0	Tummy Ache	Bali Belly
Angela	38.5	Blue Spots, Itchy	Spotty Fever
Luke	36.1		

Figure 1

The first task of analysis is to construct a framework for interpreting these examples. NIAM provides a useful heuristic which enables the expert to provide an interpretive framework:

> The Phone-Your-Sister Heuristic.
> Take the example in your hand. Pretend that you are telephoning your sister in the Himalayas. Read the example over the phone to her.

This heuristic, trivial as it may sound, gently forces the expert to make a 'visual-to-auditory' transformation. In essence, it calls upon the expert's immense subconscious talent for interpreting these *examples* and extracting their meaning, without forcing the expert to perform a detailed process of introspection or to justify her interpretation in any way.

If the expert reads out the first row in the table, the result should sound something like this:

```
The PATIENT                          ← type of entity
    with FIRST-NAME                  ← type of name (or label)
         ''Tommy''                   ← specific name (or label)
has-as-sub-lingual-temperature       ← role (or verb)
the TEMPERATURE of                   ← type of entity
         ''50''                      ← label instance
     DEGREES-IN-CELSIUS              ← type of label
and                                  ← linguistic connective
suffers-from                         ← role
the SYMPTOM                          ← entity type
    with SYMPTOM-NAME                ← label type
         ''Red Spots''               ← label instance
and                                  ← linguistic connective
suffers-from                         ← role
the SYMPTOM                          ← entity type
    with SYMPTOM-NAME                ← label type
         ''Itchy Skin''              ← label instance
and                                  ← linguistic connective
has-as-diagnosis                     ← role
the AILMENT                          ← entity type
    with AILMENT-NAME                ← label type
         ''Measles''                 ← label instance
```

This *structured verbalization* process moves from the irregular surface structure of the examples to their well-defined deep (or conceptual) structure. This conceptual structure is derived from Fillmore's linguistic theories [6].

Each sentence (fact) is simply a semantically irreducible association involving one or more entity instances. Each entity instance in the association plays a role in the sentence. In Fillmore's paper, the term *case* was used instead of role.

An entity instance is simply any distinguishable object, real or abstract, in the UoD. Entity instances cannot be directly represented (how do you represent a person?). Instead, the sentence explicitly contains label instances, which act as names for the entity instances (which appear in the sentence only through such a reference).

Consider the first elementary sentence in the interpretation above. It could be rewritten as:

```
The TEMPERATURE of
        ''50''
    DEGREES-IN-CELSIUS
is-sub-lingual-temperature-of
the PATIENT
    with PATIENT-NAME
        ''Tommy''
```

This invertibility suggests that we view a deep structure sentence as a set of (role, entity type, label type, label instance) tuples, where each role is played by precisely one entity instance in the sentence instance. For example, this elementary sentence could be formally represented as a set of tuples:

```
{ (has-as-sub-lingual-temperature, PATIENT, PATIENT-NAME, ''Tommy''),
  (is-sub-lingual-temperature-of, TEMPERATURE, DEGREES-IN-CELSIUS, ''50'') }
```

When the expert provides this structured verbalization, it is difficult not to hear the roles in each sentence, the entity types that play those roles, the label instances which are used to identify instances of those entity types, and the label types to which those instances belong.

An immediate check can be made on the quality of these well-formed deep structure sentences: can any sentences be split into smaller ones without losing information? The explicit presence of the connective "and" (or its implicit presence as a comma between clauses in a list) in any sentence is usually a good indication that it is just a conjunction of two or more smaller sentences. This can be formally checked later in the design procedure.

The interpretation of line 1, figure 1 can be split into 4 smaller, binary sentences. These smaller sentences are called elementary sentences because they are no longer splittable, without losing information. Elementary sentences need not be binary sentences, they may have any arity.

Sometimes, the expert's interpretation does not completely identify the deep structure of the sentence instance. For example

```
Tommy
has
50
```

This fact instance is not *well-interpreted* because the entity types and label types are not identified. Which label type does the name "50" belong to? Which entity type does the entity called "50" belong to?

The second quality check on the expert's interpretation detects, and corrects, this problem. The knowledge engineer must ask the following questions for each label instance in the (incorrectly structured) interpretation:

(1) What type of label is it an instance of?

The answer to this question should provide the label type that the label instance belongs to.

(2) What type of entity does it refer to?

The answer to this question should provide the entity type that the label instance refers to an entity instance of.

Finally, we must check that every possible fact instance has been extracted from the examples. For each label instance present in the original examples, the knowledge engineer asks:

(1) With which other label instance(s), does it form a meaningful sentence?

The answers to this question should reveal any fact instances that were not included in the original interpretation.

When these quality checks have been satisfied, the knowledge engineer can perform an abstraction by classification over this set of well-interpreted fact instances, to produce the most essential part of the structural section of the conceptual schema. Because the correct classifications (entity types, label types and roles) have already been provided by the expert's interpretation, this abstraction simply involves removing the label instances from the sentences. This leaves the kernel of the conceptual model:

- entity types (such as, Patient, Symptom)
- label types (such as, Patient-Name, Symptom-Name)
- roles, or verbs (such as, suffers-from, is-present-on)

Further, we can classify elementary fact instances which involve instances from the same entity types in the same roles, to form a fact type.

The next step is an immediate quality check on this abstraction by attempting to populate a database associated with the conceptual schema with all of the sentences contained in the initial examples. This will show if the abstractions have been correctly chosen or not.

It is important to distinguish strongly between the type and the instance level. Constructs at the type level are abstractions, derived by classifying constructs at the instance level.

NIAM also includes a graphical notation for both the instance level and the type concepts. This notation has been specifically developed to facilitate the effective validation of the resulting model by the expert himself. Figure 2 shows how the two deep structure sentences, or elementary fact instances, shown above can be represented graphically, in the same picture as the abstract concepts.

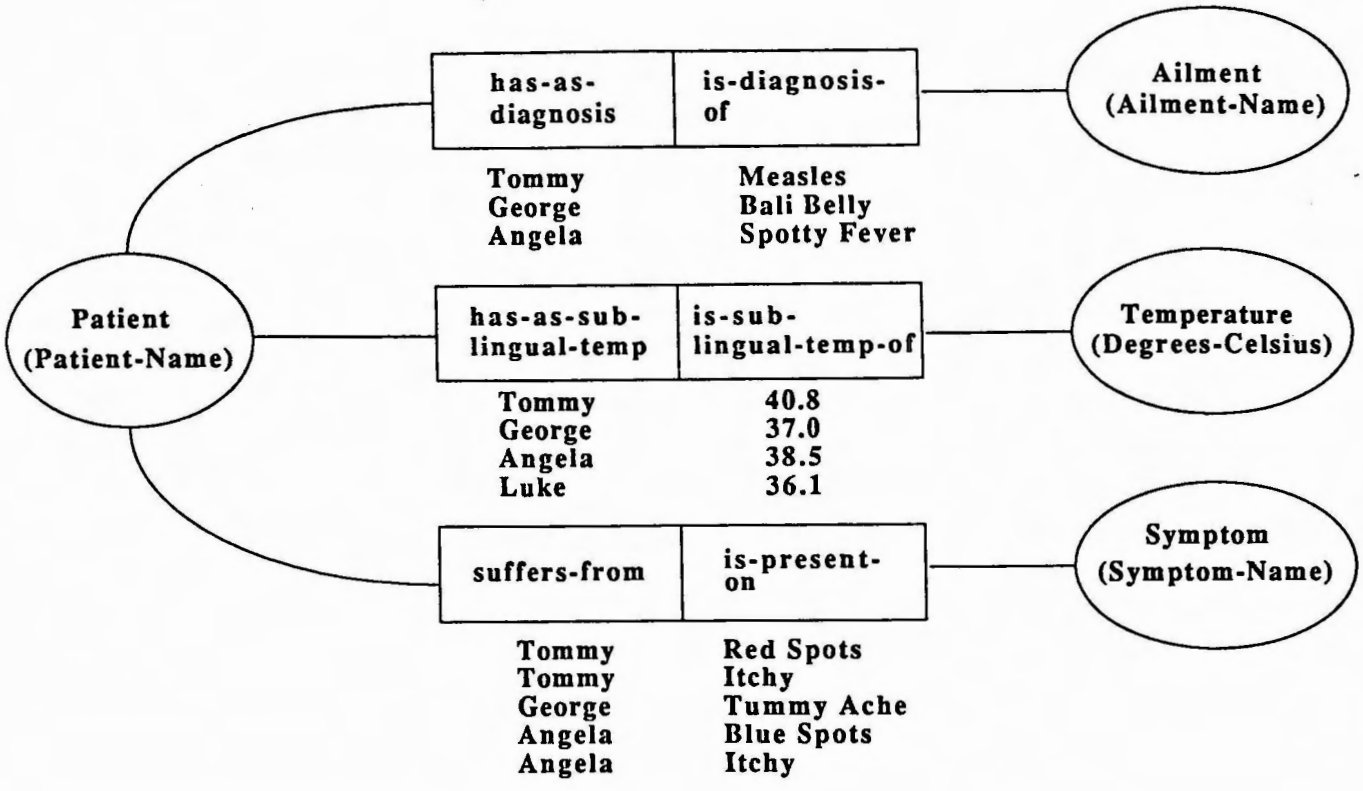


Figure 2

This graphical notation can be interpreted as follows:

- Circles represent entity types.
- Boxes represent the roles those entity types play (in a fact type).
- A set of joined role boxes represent a fact type.
- If every instance of an entity type can be uniquely identified by a corresponding instance of a single label type, then the name of the label type is shown (within parentheses) next to the name of the entity type.
- The populated table below these linked boxes lists a significant set of instances for that fact type. In this way, the diagram can be used for effective user validation, as it shows the familiar examples together with their abstractions (the type-level constructs).

We cannot emphasize the importance of this verbalization step sufficiently: it builds the major bridge between the user's familiar world of examples (the UoD) and its formal, linguistic representation (the conceptual schema) which will be used as the basis for all subsequent design steps.

The remainder of the NIAM conceptual schema design procedure consists of procedures to add constraints and inference rules to the conceptual schema, and to define any entity subtypes.

Constraints are an important part of the conceptual schema because they restrict the fact instances that are *syntactically permitted* as part of some fact type's population to those which are *semantically meaningful* within the UoD. This is necessary to bridge the semantic gap between the UoD and the formal representation of facts about that UoD, because the computer has no other formal interpretation powers. Many knowledge engineering techniques leave these constraints implicit within the data structures that they provide, or do not consider them at all.

During the analysis process, the knowledge engineer can identify the relevant constraints by formulating additional fact instances, and asking the expert the question: "Are these examples ever permitted?"

Suppose the knowledge engineer wishes to confirm (or deny) the proposed constraint that a patient may have only one temperature. This constraint (which is called a *uniqueness* constraint) can be expressed graphically as an arrow over the "has-as-sublingual-temperature" role. To validate this constraint, the knowledge engineer generates the fact instances shown in figure 3.

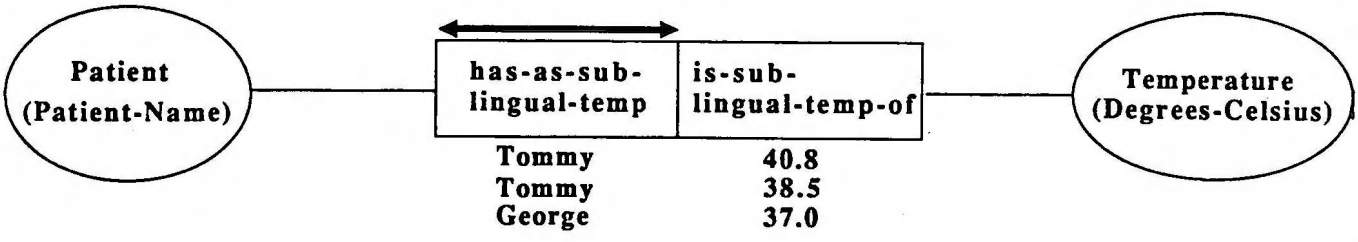


Figure 3

These fact instances are presented to the expert, with the question: "Is this ever possible?"

If the answer is YES, then the constraint is too strong, so it is denied. The uniqueness constraint must be weakened by extending it to cover both roles in the fact type.

If the answer is NO, then the constraint is sufficiently strong, and is tentatively confirmed.

This emphasis on continuous quality checks, by means of examples, is very important in NIAM, as it is the only possible way to ensure that we have modelled the *expert's* interpretation of the UoD correctly.

One of the last steps in the design procedure is to add any inference rules. An inference rule specifies how to derive the population of a designated fact type from the populations of one or more fact types. These rules must be expressed in terms of the fact types in the conceptual schema.

Rule acquisition seems to be the only step in many AI knowledge engineering techniques. Consequently, if it is difficult for an expert to express her knowledge as rules, then it will be hard to acquire that knowledge. The formal conceptual schema provides a concerete foundation for subsequent rule acquisition.

Summary. The most important aspects of the NIAM CSDP, as applied to knowledge analysis, are:

- The analysis starts with *examples* of the expert's everyday work, expressed in her natural language.

Compare this with the type-level information normally requested from the expert. In most knowledge acquisition scenarios, the expert is forced to talk about her language, rather than in it — a task she is rarely good at. Experts are experts at solving the everyday problems they encounter in their work (that is, in dealing with *examples*), not in discussing the abstractions they use to solve these problems.

These type-level statements must contain a fairly explicit specification of the entity types and label types that the expert uses to solve her problems. This means that the expert must deal

with some *very* abstract concepts in order to tell the knowledge engineer which knowledge is of relevance to her application. Experts typically find it difficult to reason at this type level, so this tends to be an unproductive procedure.

- The expert provides the interpretation, without being forced to discuss her abstractions.
- The knowledge engineer has formal rules to guide him in moving from this interpretation to an appropriate set of abstractions for a formal description.

Breuker and Wielinga et al [3] have reached a somewhat different solution. Their KADS methodology starts at a very high level of abstraction, and works towards a more concrete model, using a process of refinement.

This means that they rely on the existence of formal generic models for various classes of UoD to act as a starting point for their analysis. These interpretation models seem to be hierarchical templates which can be instantiated for the specific UoD during the analysis. They are more specific in [3]:

> An interpretation model is an abstracted conceptual model. One could imagine developing an interpretation model by deleting the domain level of the conceptual model for some particular expert task.

So, in KADS, the knowledge engineer is provided with a set of relevant abstractions that he can use to describe the UoD. This is all very well, if this UoD, or a closely related one, has been analysed before. But the important question still remains: how do we select good abstractions for a UoD? We must start somewhere!

The expert must know a good set of abstractions, in order to be able to work effectively with problems in that UoD. But most experts find it very difficult to state these abstractions, or to justify them. On the other hand, the knowledge engineer is rarely an expert in the UoD being formalized, so he cannot provide these abstractions.

NIAM resolves this dilemma, by using a heuristic which enables the expert to implicitly identify the correct abstractions, without requiring her to explicitly articulate this information. The knowledge engineer constructs the conceptual model by a simple process of syntactic abstraction over this interpretation framework.

There is a general problem with refinement (or hierarchical decomposition) methodologies (such as KADS) because they lack any clear signposts indicating which object to decompose next, which subobjects to decompose it into, and "red and green traffic lights" indicating when to stop decomposing an object. Hierarchical descriptions are convenient for explanation and documentation, but their associated methodologies typically lack the precise guidance required for effective analysis and design.

Such methodologies are descriptive, rather than prescriptive. They can describe what the end-result should look like, but they cannot give the knowledge engineer proper (formal) guidelines on how to produce this end result.

NIAM starts from the available examples, and works toward an abstract model, using a formally-guided process of abstraction. At all times, the knowledge engineer has explicit rules indicating which design technique to apply next, and how to validate the result of applying that technique.

- The expert validates the conceptual model.

As a general principle, the expert *must* be involved in the validation process because the conceptual model being validated must reflect the expert's interpretation of the UoD.

NIAM supports continuous quality checks. Most other knowledge analysis methodologies wait until the prototyping stage before they perform any validation. Interestingly, this is often the first stage when examples become available again. The knowledge engineer must redesign and reimplement to correct errors.

3. KNOWLEDGE REPRESENTATION: TRANSFORMATION RULES FOR KEE

KEE [5] is Intellicorp's proprietary Knowledge Engineering Environment. KEE's knowledge representation component includes units (or frames), slots, facets, links and values.

Here is an example of a unit:

```
UNIT Tommy
  MEMBER OF: Patients
  SUPERCLASSES: ()
  OWN SLOT suffers-from
      Values: (RedSpots ItchySkin)
      ValueClass: Symptoms
  OWN SLOT has-as-sublingual-temperature
      Values: (50)
      ValueClass: Integer
      Cardinality.Max: 1
      Cardinality.Min: 1
  OWN SLOT has-as-diagnosis
      Values: (Measles)
      ValueClass: Ailment
      Cardinality.Max: 1
      Cardinality.Min: 1
```

This unit contains three slots, both of type "OWN". It has a member link to the unit named Patients. It has no superclass links. The member link is intended to represent the type-instance relationship. The superclass link is intended to represent the type-subtype relationship.

KEE units are used to represent *both* entity types and entity instances, in the UoD. There is no syntactic distinction in KEE between units which represent types and units which represent instances. Further, there is no way to restrict, using KEE, which links a unit may participate in. This makes it difficult to provide a consistent semantic interpretation for any KEE data structure, and as a consequence, impossible to provide a prescriptive methodology for representing knowledge in KEE.

One solution is to restrict the ways in which we use KEE, by adopting a rigid (self-enforced) design discipline. To resolve this semantic crisis, we agree to represent only a single metalevel (or level of abstraction) in any KEE knowledge base. This means that there will be only one level of member links in the data structure, at the lowest level. Further, we will agree that every entity must be an instance of exactly one type. Therefore, every unit which represents an instance will have a member link joining it to precisely one unit, which must represent a type.

This means that we can infer, from the links associated with a unit, whether it represents an entity instance or an entity type. If the unit has a member link connecting it to one other unit, then it represents an entity instance. If the unit has no member links connecting it, or more than one member link connecting it (it must be at the head of all these links), then it represents an entity type.

Further, there is more than one way to represent an entity. Some entities may be represented as units. Others may be represented as facet values. Some may even be represented in both ways at once.

The representation chosen for the entities involved in a fact, determines the way in which the fact will be represented.

When the first entity is represented by a unit, and the second entity is represented by a value, then the fact must be represented by placing that value in the appropriate slot in that unit.

When both entities are represented by units, the fact can be asymmetrically represented by placing the name of one unit, as a value, in the appropriate slot in the other unit. As a consequence, one of the entities will now have a dual representation (as a unit and as a value). There is no semantic basis for choosing which unit should contain the slot.

Alternatively, the fact can be symmetrically, and redundantly, represented by a value in the appropriate slot in each unit. However, KEE provides no declarative, centralised way to maintain mutual consistency between the two slot values. Hence, there is no way that the system can prevent an update anomaly from occurring. The knowledge base designer must explicitly control the consistency of these slot values by writing LISP update procedures.

Finally, if both entities are represented only by values, then there is no way to represent the fact in KEE.

As you can see, there are too many different (and, in many cases, incompatible) ways of encoding facts in KEE data structures. So far as representing declarative facts is concerned, KEE shares some of the problems of the "multi-target" fact representation data models of the 1970s (namely, CODASYL DDL). This analogy is sufficiently strong to suggest that knowledge engineers using KEE will have certain difficulties in designing, maintaining and integrating KEE knowledge bases, similar to the problems that database designers had in designing, maintaining and integrating CODASYL DDL databases. One of the most severe of these problems is the need for extensive reprogramming (or rewriting of inference rules) whenever a minor change is made to the underlying data structures. Many of these changes do not even affect the semantics of the knowledge base, because they simply change from one way of representing a fact to an alternative way of representing the same fact. For more detail on the semantic anomalies of KEE, and their consequences, refer to [18].

Therefore, we need rules to instruct the knowledge engineer in which way to encode which facts. If these facts are represented in a conceptual model, then these rules must translate the conceptual model to an equivalent (so far as the semantics of KEE permit) frame representation. This is not to suggest that this translation is all that is needed for knowledge acquisition: however it forms a necessary foundation for further tasks.

A naive mapping from facts in a NIAM conceptual schema to units and slots in KEE is as follows:

(1) For every entity type, create a corresponding unit.

(2) For every fact type, create a member slot in *each* of the units representing the entity types in that fact type.

For example, the conceptual schema in figure 2 would be mapped to the following KEE units:

```
UNIT Patients                                    UNIT Symptoms
  MEMBER OF: ()                                    MEMBER OF: ()
  SUPERCLASSES: ()                                 SUPERCLASSES: ()
  MEMBER SLOT suffers-from                         MEMBER SLOT is-present-on
         Values: ()                                       Values: ()
  MEMBER SLOT has-as-sub-lingual-temperature
         Values: ()
  MEMBER SLOT has-as-diagnosis
         Values: ()

UNIT Ailments                          UNIT Temperatures
  MEMBER OF: ()                          MEMBER OF: ()
  SUPERCLASSES:()                        SUPERCLASSES: ()
  MEMBER SLOT is-diagnosis-of            MEMBER SLOT is-sub-lingual-temperature-
         Values: ()                             Values: ()
```

Similar rules can be used to create frames for the fact instances that populate the fact types in the conceptual schema.

(1) For every entity instance, create a corresponding unit, and create a member link between that unit and the unit representing that entity's type.

This is impractical for large numbers of instances. Hence the traditional distinction between a database and a knowledge base: the database has many instances of a few types, while the knowledge base has a few instances of many types.

(2) KEE's inheritance procedures will create own slots, in each unit which represents an entity instance, corresponding to each member slot in the unit representing that entity's type.

    These own slots will be used to store fact instances involving that entity instance.

(3) For each fact instance: place the name of the unit, which represents the first entity instance, in the slot, corresponding to the fact type, in the unit, corresponding to the second entity instance.

    Then place the name of the unit which represents the second entity instance, in the appropriate slot of the unit which represents the second entity instance.

The unit representing the entity instance named "Tommy", together with the fact instances recorded about that entity instance, in figure 2, is:

```
UNIT Tommy
  MEMBER OF: Patients
  SUPERCLASSES: ()
  OWN SLOT suffers-from
      Values: (RedSpots ItchySkin)
  OWN SLOT has-as-diagnosis
      Values: (Measles)
  OWN SLOT has-as-sublingual-temperature
      Values: (50)
```

Of course, this is a very trivial mapping. It creates a number of units that are, strictly speaking, unnecessary and it records every fact twice (once with each unit that represents an entity involved in the fact). This ensures a symmetric representation of each fact, but at the cost of

redundancy, and the risk of update anomalies.

In many cases, it may not be desirable to create units for the entity instances of Measles, RedSpots and ItchySkin because the only facts recorded in those units are also recorded elsewhere (in the unit Tommy).

4. CONCLUSIONS

Knowledge analysis, as reflected by current practice in the expert systems community, seems to be an artistic discipline, rather than an engineering one. Of course, it is very hard for amateurs to begin practicing an artistic discipline, and it often requires divine inspiration in order to succeed. We can improve this situation by reducing the knowledge engineer's "artistic freedom", and replacing general descriptive guidelines with a set of formal design techniques, together with explicit rules telling the knowledge engineer when to apply each technique.

By applying a well-guided information analysis methodology, such as NIAM, to the task of knowledge analysis, it is possible to start with examples of the information that the expert deals with in her day to day work, and proceed to develop a conceptual model of the UoD, relying on the expert to provide the interpretation framework, but without forcing her to explicitly articulate the abstractions she uses, and resulting in a high-quality conceptual schema which can be transformed into the required representational data structures (such as KEE frames).

5. REFERENCES and BIBLIOGRAPHY.

1. J. H. Alexander, M. J. Freiling, S. J. Shulman, S. Rehfuss and S. L. Messick, Ontological Analysis: An Ongoing Experiment, *Proceedings AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems, Banff, Canada*, 1986.
2. J. Breuker and B. Wielinga, Use of Models in the Interpretation of Verbal Data, in *Knowledge Elicitation for Expert Systems: A Practical Handbook*, A. Kidd, (ed.), Plenum Press, New York, 1987.
3. J. Breuker, B. Wielinga, M. Van Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, J. Billault, M. Davoodi and S. Hayward, Model-Driven Knowledge Acquisition: Interpretation Models, Esprit Project 1098, Deliverable Task A1, Memo 87.
4. E. Falkenberg, Foundations of the Conceptual Schema Approach to Information Systems, in *Lecture Notes of the NATO Advanced Study Institute on Database Management and Applications, June 1-13, 1981, Portugal*, North-Holland, 1982.
5. R. E. Fikes and T. Kehler, The Role of Frame-Based Representation in Reasoning, *Comm. ACM 28*, 9 (Sept 1985), 904-920.
6. C. J. Fillmore, The Case for Case, in *Universals in Linguistic Theory*, E. Bach and R. T. Harms, (eds.), Holt, Rhinehart and Winston, Inc., 1968.
7. M. Freiling, J. Alexander, S. Messick, S. Rehfuss and S. Shulman, Starting a Knowledge Engineering Project: A Step-by-Step Approach, *AI Magazine*, Fall 1985, 150-164.
8. D. A. Jardine, Semantic Agreement and the Communication of Knowledge, in *Proceedings of IFIP Conference on Data Semantics (DS-1)*, T. B. Steel Jr and R. Meersman, (eds.), Elsevier (North Holland), 1986, 71-82.
9. W. Kent, *Data and Reality*, North-Holland, 1982.
10. R. Meersman, Knowledge and Data: A Survey in the Margin of the IFIP DS-2 Conference, in *Entity-Relationship Approach*, S. Spaccapietra, (ed.), North-Holland,

1987.

11. D. Michie, Inductive Rule Generation in the Context of the Fifth Generation, *Proceedings of the International Machine Learning Workshop*, 1983, 65-70.
12. K. Morik, Acquiring Domain Models, *Proceedings AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems, Banff, Canada*, 1986.
13. G. M. Nijssen, A Framework for Advanced Mass Storage Applications, *Proceedings IFIP Medinfo 1980, Tokyo*, 1980.
14. G. M. Nijssen, *Knowledge Engineering, Conceptual Schemas, SQL and Expert Systems: A Unifying Point of View*, Dept. Computer Science, University of Queensland., 1986. Research paper. Also available from NOVI (Dutch National Institute for Informatics), Netherlands, 1986..
15. G. M. Nijssen, On Experience with Large-Scale Teaching and Use of Fact-based Conceptual Schemas in Industry and University, in *Proceedings of IFIP Conference on Data Semantics (DS-1)*, R. Meersman and T. B. Steel Jr, (eds.), Elsevier North-Holland, 1986.
16. G. M. Nijssen and T. A. Halpin, Fact-Based Conceptual Schema and Relational Database Design Using NIAM , Forthcoming , 1988.
17. S. Regoczei and E. P. O. Plantinga, Ontology and Inventory: A Foundation for a Knowledge Acquisition Methodology, *Proceedings AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems, Banff, Canada*, 1986.
18. S. Twine, Representing Facts in KEE's Frame Language, *Proceedings IFIP WG2.6/WG8.1 Conference on the Role of Artificial Intelligence in Databases and Expert Systems, Guangzhou, China*, June, 1988.
19. *Concepts and Terminology for the Conceptual Schema and the Information Base*, J. J. van Griethuysen, (ed.), 1982. Report of ISO TC97/SC5/WG5.

Modelling in KBS Development*

Guus Schreiber†
Joost Breuker
Bert Bredeweg
Bob Wielinga

Department of Social Science Informatics
University of Amsterdam, The Netherlands

Abstract

There is an increasing need for a methodological approach to the development of knowledge-based systems. Esprit Project P1098 (KADS) is concerned with the development of such a methodology. The basic idea behind this methodology is to view the process of KBS development as a *modelling* activity. Two types of models are constructed in this process: a model of the real world domain (analysis model) and a model of the artifact (design model). In this paper the focus is on the structure of these models.

Keywords: epistemological models, modelling language for problem solving, modelling language for KBS design, KBS development methodology, KBS life cycle.

Introduction

A knowledge based system (KBS) is a model of some expertise. Like all modelling activities, KBS development 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. Modelling is a complicated form of design. In design a structure is generated that satisfies constraints. In modelling the constraints are more severe in the sense that some form of correspondence should be maintained between the data and the structure. The type of correspondence is dependent on the purpose of the model.

The objective 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. The problems that have to be 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. Knowledge acquisition is an activity that is not, or only in a rudimentary form, required in conventional software development, where requirements are the major object of analysis.

In expert tasks the decomposability (analysis) of a particular problem 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' from the 'variations' in

*The research reported here was carried out in the course of a research project partially funded by the Esprit Programme of the Commission of the European Communities as project number 1098. The partners in this project are STC Technology Ltd., SCICON Ltd., KBSC, (all UK), SCS[GmbH (W-G), Cap Sogeti Innovation (F), and the University of Amsterdam (NL).

†Correspondence Address: Department Of Social Science Informatics, University Of Amsterdam Herengracht 196, 1016 Bs Amsterdam, The Netherlands, Tel. +31 20 525 9111, Electronic mail: guussc@swivax.UUCP

these problems. The required abstractions are called models. A modelling language consists of (1) definitions of elements and (2) structuring principles for building models. A modelling language is a tool for using adequate abstractions in the development of knowledge-based systems. Whether an abstraction is adequate, can be argued from a theoretical and from a empirical point of view. In this paper the theoretical considerations are emphasised, but practical experience with these tools over the last four years, both within the KADS project and in commercial applications outside this project, has provided ample empirical justification and correction. [11].

1 Models in the Development Process

1.1 The KADS Methodology

Esprit Project P1098 (commonly referred to as the KADS project) is concerned with the development of a methodology for the construction of knowledge-based systems (KBS). The KADS methodology currently offers the knowledge engineer three types of support:

1. *Modelling Languages*
   A basic idea of the methodology is that the process of KBS construction is viewed as a mapping of model of the real world onto a model in an artificial world. The model of the real world is a model of the problem solving process in the KBS domain, as observed with domain experts or other information sources. The model of the artificial world is the model of the KBS to be developed. The methodology offers descriptive languages for these two models.

2. *Life Cycle*
   The process of KBS construction is partioned in a number of phases. The methodology is currently concentrated on the first two phases: analysis an design. Within these phases a number of tasks are recognised, that should be carried out by the knowledge engineer.

3. *Software Support Tools*
   A set of software tools has been developed that supports the knowledge engineer in applying the methodology. The current tool set is restricted to the analysis phase.

In this paper we focus on the model description languages of the methodology. An overview of the current status of the life cycle can be found in [2]. The software tools are discussed in [1]. [10] gives an overview of the methodology as a whole.

1.2 Models in KADS

Figure 1 illustrates the modelling in KADS. The X-axis represents the level of abstraction; the Y-axis the 'number of transformations' to turn the data into a system.

The figure uses Brachman's notion of describing 'semantic network' models in five levels [4]. The levels of Brachman in terms of KBS development can be characterised as follows. At the linguistic level the (verbal or other) data are 'cleaned' in terms of the identification of concepts. At the conceptual level concepts are classified. At the epistemological level the uncovering of the structuring principles takes place. The epistemological level is the highest level of abstraction. At the logical level the appropriate formalisms for which implementation realisations are known are selected and assembled. The implementation level consists of the application of programming constructs.

The models involved in the development process of a KBS are shown in figure 1 in relation to these five levels. DATA in this figure consists of the raw input data from the real world: text books, interviews, thinking aloud protocols, etcetera. DATA is at a low level of abstraction and can easily be mapped onto the linguistic level of Brachman. M1 is a model of the problem solving process at both the conceptual and the epistemological level. It is a model of the domain expertise

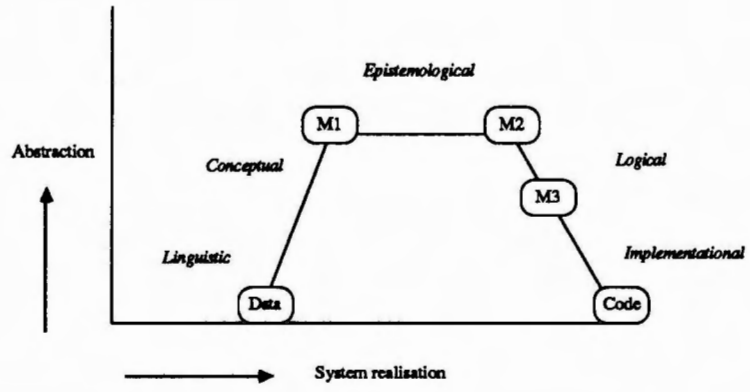


Figure 1: Models in KBS Development

and is called the *conceptual model.* M1 is real world oriented in the sense that it only describes the competence in expert problem solving, and has a cognitive flavour. In particular, it does not take issues with regard to the artifact into account, i.e. it is completely neutral with respect to implementation formalisms.

M2 – i.e. the 'design' model – is a model at the same level of abstraction, but of the artifact rather than of the real world. It is, in fact, a high level system design. The idea is that the M2 model can be transformed into a detailed system design (M3) and subsequently into actual code without major decisions having to be made.

Two points can be raised with regards to this figure:

1. Why are these intermediate models at a high level of abstraction necessary? Is it not possible to move directly from DATA to M3 and/or implementation?

2. Why is it necessary to distinguish between M1 (the conceptual model) and M2 (the design model of the artifact)?.

A direct mapping of DATA onto M3 is in fact a kind of rapid prototyping. A knowledge-based system development environment (a 'shell') usually acts in such an approach as a template of the M3 model. The knowledge engineer uses the primitives of the environment to represent the DATA elements in the artifact. There are several problems with this approach. First, the mapping from DATA onto M3 is very difficult. The knowledge engineer has to bridge a large gap. This is what constitutes the well known 'knowledge acquisition bottleneck'. There is a large amount of data, which may have a large variety of interpretations and which provide descriptions of expertise at various levels and in various varieties. Therefore, in complex domains a direct mapping is hardly ever possible.

Another problem is posed by the observation that the resulting prototype usually remains a prototype: i.e. it cannot be expanded or modified to a full-scale KBS. And also, even if the prototype is a useful system by itself, a small change in requirements is likely to cause problems.

Yet another problem is that knowledge-based systems developed with rapid prototyping are difficult to maintain and/or enhance. This is due to the the lack of explicit high level design decisions. Often the system is more tuned to the possibilities of the environment used than to the needs of the problem solving domain (see also [6]). An epistemological description is an important aid in the development of a maintainable knowledge-based system. Moreover, such a high level description can also improve the system's explanation possibilities [7,13].

M1 is a competence model of the problem solving as observed in, for example, domain experts. The major difference between M1 and M2 is that M2 takes into account external requirements (*e.g.* speed, software) and user requirements (*e.g.* a natural language interface). These requirements

may limit in certain instances the scope of the conceptual model M1, but they usually do not change the contents of this model. We have experienced that building an M1 model without having to worry about system requirements makes life easier for the knowledge engineer.

The distinction between M1 and M2 is also useful, if one opts for another problem solving method in the artifact than the one used by the domain expert. For example, if the domain expert uses heuristics to find a particular solution, one may choose to use a *generate and test* method in the KBS, because its computational resources are larger. This situation occurred in one of the experimental KBS developments within P1098 [3]. The distinction between M1 and M2 can in that case be used to explain the differences in behaviour between the system and the expert.

2 Conceptual Models: what's in expertise

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 reliably solvable, from less reliably or even 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'.

Developing a KBS involves modelling of expertise, *i.e.*, the structuring of the required knowledge, inference competence and (flexible) use of knowledge. This model is the *conceptual model.* The term *Conceptual model* denotes the model the designer of a computer system (KBS, conventional software) has in mind and/or specified when implementing a system. The role of a conceptual model is a high level functional specification that is used in designing and implementing a KBS.

Flexibility refers to the control of complexity. Current views consider the distinction between domain knowledge and "meta knowledge" to control the reasoning at the domain as a viable, but problematic solution [7]. This solution is problematic for various reasons: it is computationally inefficient, and moreover, there is no theoretical boundary to prevent proliferation of meta-levels. In the framework we propose, we distinguish four layers of describing expertise: each level has a different functionality, and, moreover, it seems that the fourth level is a top-level: not in a formal sense, but in a psychologically valid sense.

A summary of these four layers is presented in table 1.

Layer	*Elements*	*Relation with lower layer*
Strategy	Plans, meta rules	
		controls
Task	Goals, task structures	
		applies
Inference	Meta classes, knowledge sources	
		describes
Domain	Concepts, relations	

Table 1: Layers of a Conceptual Model (after [15])

The first layer contains the static knowledge of the domain: domain concepts, relations and complex structures, such as models of processes or devices. First generation expert systems contain only descriptions of expertise at this level: often incomplete, because the domain knowledge may not only consist of knowledge to perform specific tasks, but also knowledge that contains underlying principles and justifications (*i.e.*, 'support' knowledge [7], or: 'domain theory'). Formalisms for

describing domain knowledge are the classical knowledge representations, which distinguish *e.g.,* concepts and relations.

The second layer is the inference layer. It describes the inference making competence that is required for solving a problem based upon the domain knowledge. It states *what inferences* can be made: not how or when they are made. The how is dependent on the structures in the domain layer that allow for methods to be applied. For instance, if at the inference layer some classification competence is specified, this may be based on is-a structures in the domain knowledge, or may be ad-hoc, based upon a number of heuristic classification rules. The 'when' is specified at the next, task layer (see below). The inference structure provides a flow of data view; the task structure a flow of control view superimposed on this inference structure.

The "vocabulary" for the inference layer consists of three elements: knowledge sources, metaclasses and dependencies between these. A structure of these elements is called an inference structure (cf. [8]). This layer marks the main difference between the KADS description and other meta-level reasoning descriptions. The inference layer is an intermediate competence description. The domain knowledge is thus not directly controlled by a flexible task description or strategic reasoning components. The inference level is discussed in more detail below.

2.1 Inference layer: knowledge sources and metaclasses

A knowledge source can be defined as a primitive inference making function, in which the metaclasses are the roles or input/output parameters of this function. Metaclasses stand for a set of domain concepts and point to the roles these domain concepts play in the problem solving rocess. Well known examples of those roles are: hypothesis, evidence, factor, diagnosis, plan, etc. Distinguishing the role of a concept in some problem solving or reasoning process from what the concept "means" is important. It provides an almost orthogonal view on domain structures in the sense that some domain concept may play various roles. For instance: the concept that is a hypothesis (*e.g.,* a disease) at a certain moment in the problem solving process may turn out to be the diagnosis, because it is supported by evidence. These dependencies between terms like hypothesis, evidence and diagnosis are not accidental: they form a pattern in reasoning. Metaclasses are "handles" which mediate between domain knowledge and control, or strategic reasoning.

A knowledge source (KS) identifies a type of inference. An inference is an operation on some input state producing a new, output state. The relation between the input state and the output state denotes the type of inference and is expressed by an action term like 'assemble' or *specify.* A KS is an abstraction of the inference process, that only describes that an inference can be made. How it is made can be specified as a method that is applied to the domain knowledge, but it appears to be more practical to specify the precise method in the design model (M2) (see next section).

The typology of KS' s – or inferences – is based upon the epistemology of concepts. Inferences are operations on concepts, so knowledge sources can be viewed as abstracted operations (functions) on abstracted concepts, *i.e.,* their naked structures, not filled with their particular types. This is what is meant by the fact that the inference layer is a meta-level of the domain level. KS's can be classified according to which elements of the structure of a concept, or structures of concepts it manipulates. In other words, inferences can be typed in terms of "what can happen to a concept". A concept can be either part of a description (say: generic knowledge) or of a set of instances.

In table 2 an overview is presented of KS as identified up till now (for further details: see [5], chapter 5.1.2.). Between brackets the transition between the types of knowledge is summarised. This typology is probably not exhaustive, but in practice it has been sufficient to model a large variety of domains.

Knowledge sources can be viewed as functions in the problem solving process; the metaclasses are the arguments or *roles* in this process. A metaclass has therefore no structure; it is a slot that can be filled with a domain concept (*i.e.,* its actual reference to the domain layer). Because roles are dependent on the stage in the problem solving process a typology of metaclasses is difficult

Group	*Operation*	*Arguments*
Generate concept	instantiate	description → instance
	classify — identify	instance → description
	generalise	set → description
	abstract	description → description
	specify	description → description
Differentiate between concepts	compare	values
	match	structures
Structure manipulation	assemble	set → structure
	sort	set → set
	decompose	structure → set
	transform	structure → structure
Change concept	assign-value	compute

Table 2: Typology of Knowledge Sources

to construct, because the structure of the problem solving process (*e.g.*, inference structure and task structure) varies. Therefore, we have only used a very crude leading principle in viewing the problem solving process in a pseudo temporal/causal way.

An important reason for causing much problems in the typing of metaclasses is that the vocabulary in our natural languages contains only few terms to denote roles in reasoning. Therefore, many metaclasses have a newspeak flavour as a consequence of a lack of a well structured naive cognitive psychology in our society. In figure 2 this provisional typology of metaclasses is presented.

Knowledge sources and metaclasses and their dependencies form inferences structures. A simple example of an inference structure is presented in section 2.3. -Example of a Conceptual Model.

2.2 Control: the task and strategy layers

The third layer is the task layer. At this level the basic objects are goals and tasks. Tasks are ways in which knowledge sources can be combined to achieve a particular goal. A task structure is a fixed strategy that controls the use of KSs and the interactions with the environment/user. The ingredients for task structures are *goal statements* and *control statements*. Goal statements are specified as an action term and one or more objects, *e.g.*, 'obtain (data)'. At the lowest level a goal statement may refer to the invocation of a knowledge source. Goal statements may consist of structures of (sub)goal statements.

Although models of fixed strategies are very powerful in problem solving, they do hardly account for the flexibility of expert behaviour. Flexibility in strategic reasoning would require a planning component that dynamically generates a goal tree. Such a planner would use knowledge about general problem solving methods, such as means-end analysis, heuristic classification, etc., and adapt its planning on the basis of interactions with the user and/or monitoring its own progression in the course of the problem solving process.

A full description of competences at the strategic level is impossible. The strategic level represents "problem solving and knowledge acquisition *abilities*", or "intelligence" in the psychometric sense. Strategic reasoning is of utmost importance in attacking new types of problems, but for expert -*i.e.*, highly overlearned- problem solving skills, strategic reasoning compiles out in a semi-fixed task structure, which varies over known types of problems. Thus in practice a full description of strategic reasoning is not required for building an expert system. The strategic level in the conceptual model is therefore only specified by some rules which often have the form that if some type of problem occurs, the task structure should be modified in a particular way. Only if we want

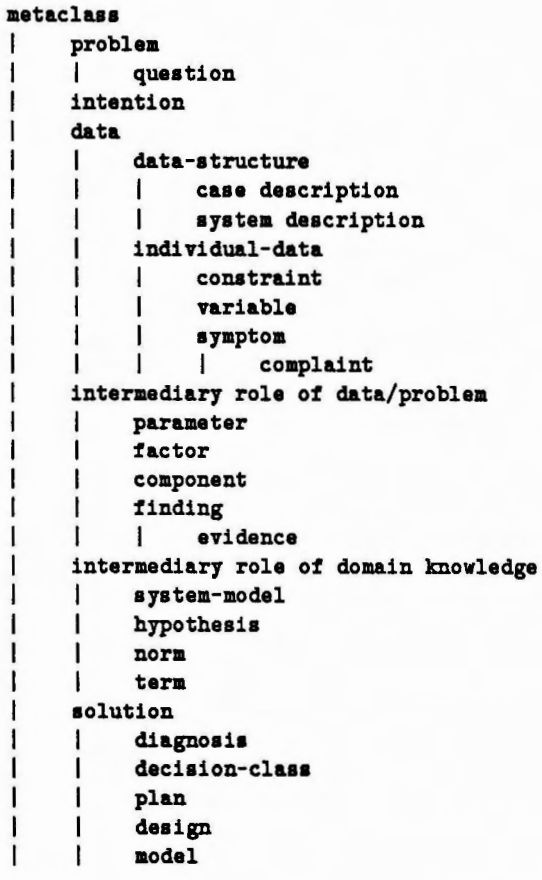


Figure 2: Typology of metaclasses (tentative)

to have a highly flexible and intelligent system which will acquire knowledge and skills for new types of problems (*e.g.*, in new, adjacent domains), the specification of the strategic level becomes essential. This is obviously far beyond the state of the art. In the simple example of a conceptual model in the next section no specifications of the strategic level occur, because such a degree of flexibility is not required.

2.3 Example of a conceptual model: the Flora expert

The example of a conceptual model presented here is a very simple one compared to the real life models that have been developed (*e.g.*, [9]). However, the description of a real life example would be more than a conference paper by itself.

Suppose we have the following goal for a KBS:

> The system should be able to perform the task of a biologist, who can classify plants by obtaining characteristics of a certain plant and apply the rules of the flora to find the correct species.

The conceptual model is specified in an abbreviated form as follows.

2.3.1 Domain Knowledge

- entities (representing plants)
- attributes and values (representing defining attributes for a class)
- subsume relations (relation between entities)
- heuristics for attribute selection

2.3.2 Inference Layer

Figure 3 shows a graphical representation of the inference structure.

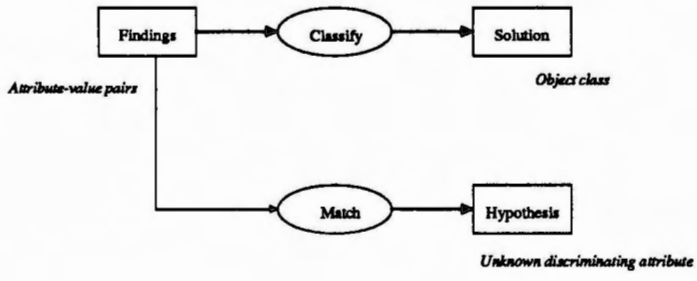


Figure 3: Inference structure of the Flora Expert

2.3.3 Task Structures

```
Classify Plant
   obtain-data(object attributes)  (Goal)
   classify(object attributes)  (Knowledge source)
   IF solution
   THEN
       display(object class)  (Goal)
   ELSE
```

```
        select-request( attribute ) (Knowledge Source)
        display-request (Goal)
        Classify Plant  (Goal)

Explain
   display(trace(classify))  (Goal)
```

3 Design Model (M2)

The design model is essentially a transformation of the model of the problem solving process, on the basis of the external and user requirements. The design model reflects three viewpoints, describing respectively the function, the behaviour and the structure of the artifact.

Figure 4 gives an overview of the design process The boxes represent the elements of the modelling language. The top of the figure depicts the *analysis model*, which consists of the *conceptual model* and the *external (and user) requirements*. It is the input for the overall design process. The different steps in the design process are, in this picture, represented as ovals.

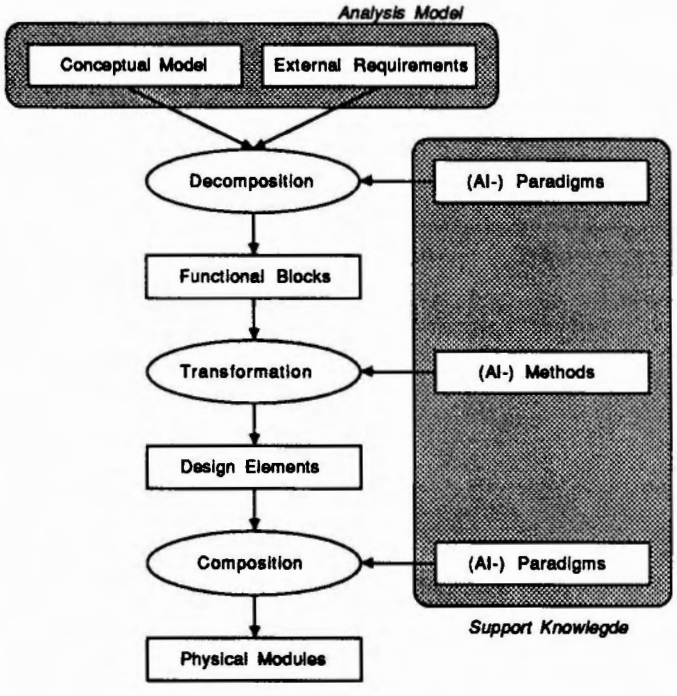


Figure 4: The Design Process

3.1 Design Desciptions

Table 3 gives an overview of the descriptions in the design model.

The first description is the *functional description*. The functional description should encompass a specification of the internal and external functionality of the perceived system. Or to put it differently, it should specify *what* is required of the final artifact.

Viewpoint	*Elements*
Functional	functional blocks, consist-of relations, I/O relations, control relations
Behavioural	methods, design elements
Structural	physical modules, architecture, environment

Table 3: Elements in the Design Model

The functional description consists of a conversion of the conceptual model into a set of functional blocks. During this process the system requirements are integrated. Each functional block represents a distinct functional unit of the final artifact and can be classified according to a number of features. The most relevant features are the *function type* and the *relations with other functional blocks*. The function type is a characterisation of the role that the function plays in the perceived system (*e.g.* problem solving, data storage). The relations of a functional block with other functional blocks are *consist-of* relations, *input-output* relations, and, *control* relations. These relations can be used to specify that: a functional block can be decomposed into other functional blocks; a functional block can use output from, or provide input for, other functional blocks; a functional block may control other functional blocks. An example (partial) functional description of knowledge-based system which classifies plants is given in figure 5.

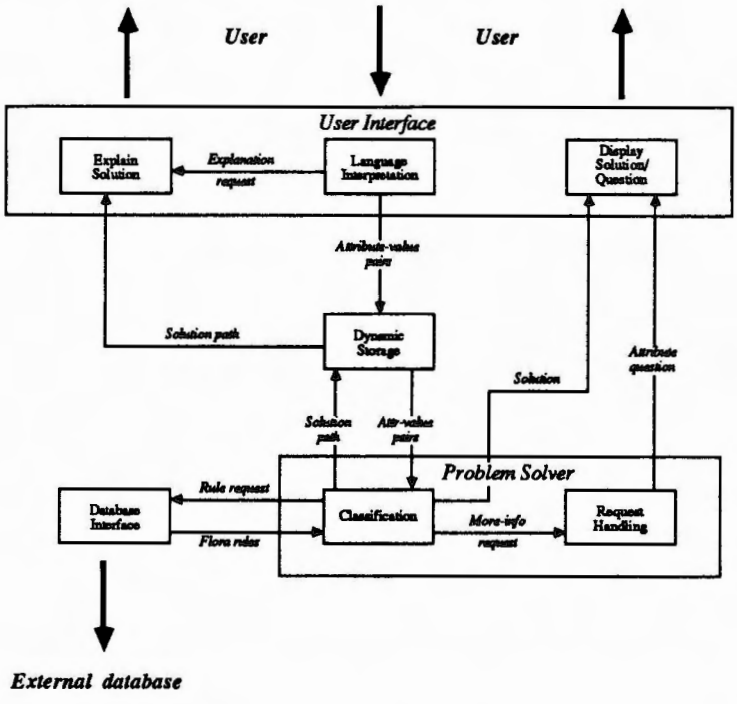


Figure 5: Partial functional description of a Flora Expert

The second viewpoint constitutes a *behavioural* description. The behavioural description consists of a set of selected *methods* and their corresponding *design elements*. Methods specify the way in which the functions are realised by the artifact and usually refer to AI methods like: search

algorithms, classification using isa-relations, and others. Methods should be seen as abstract entities, associated with a set of design elements, which are the physical parts that make up the method. The selection of (AI-) methods is probably the most significant difference between KBS development and conventional system design. Some example methods and their design elements are listed in table 4.

Method	*Function Type*	*Design Element*
Hierarchical refinement	Problem solving	Classifier, subsumption relations, class definitions, attribute value pairs
STRIPS Planning	Problem solving	Operators (add-list, delete-list, precondition), propositions, operator selector, subgoal administrator
ATN parsing	Data I/O	Parser, ATN grammar, lexicon, string, parse tree
ATMS	Data storage	Scheduler, nodes (datum, labels, justifications), consumers

Table 4: Methods and their Design Elements

The third viewpoint describes the structure of the artifact. In this structural description the design elements of the chosen methods are aggregated into physical modules. The objective is to provide the basis for detailed design and implementation. Or to put it differently, the structural description specifies *how* the functional requirements at the functional layer can be addressed in the final artifact. The structure of physical modules represents the *architecture* of the artifact. Figure 6 shows the architecture of the example Flora Expert.

The AI methods selected in this case are hierarchical refinement (realising the classification function), a production system (to realise the request handling function), and ATN parsing (to realise a natural language interface). Notice in this figure the different organisational principles in comparison with the functional description.

In the structural description the notion of *environment* is introduced. An environment is an off-the-shelf software system 'containing' a set of methods and the corresponding design elements. Examples are: EMYCIN KEE, and Prolog. Environments can be divided into *open* and *closed* environments. In a closed environment the set of methods and design elements is not expandable, whereas an open environment has possibilities for expanding the set of methods and design elements. Open environments can be further specified as being *weak* or *strong* environments, depending on the size of the set of pre-defined methods and design elements. Prolog can in this sense be viewed as an example of a weak environment, whereas KEE can be seen as an example of a strong environment. In the case of a closed environment the physical module structure of the artifact is largely fixed. Closed environments can thus be seen as *skeleton architectures* of the artifact. A special category of (closed) environments are *shells*. A shell is an environment, containing a set of methods and corresponding design elements necessary to carry out a specific task. The shell is a complete system except for the domain dependent part of the design elements. This domain dependent part has to be filled in for a particular target KBS.

3.2 Support for the Design Process

To guide the overall process of converting the analysis model into the design model, a paradigm has to be selected. A paradigm refers to a principle, or a set of principles, which is based on a theory or a methodology. An example of such a paradigm, comming from the field of structured system design, is that of *minimising coupling* and *maximising cohesion* [16]. However, only using conventional software engineer paradigms will not be sufficient for the development of knowledge based systems. For that purpose *AI-paradigms* are needed which take different aspects of the *knowledge*, that is going to make up the final artifact, into account.

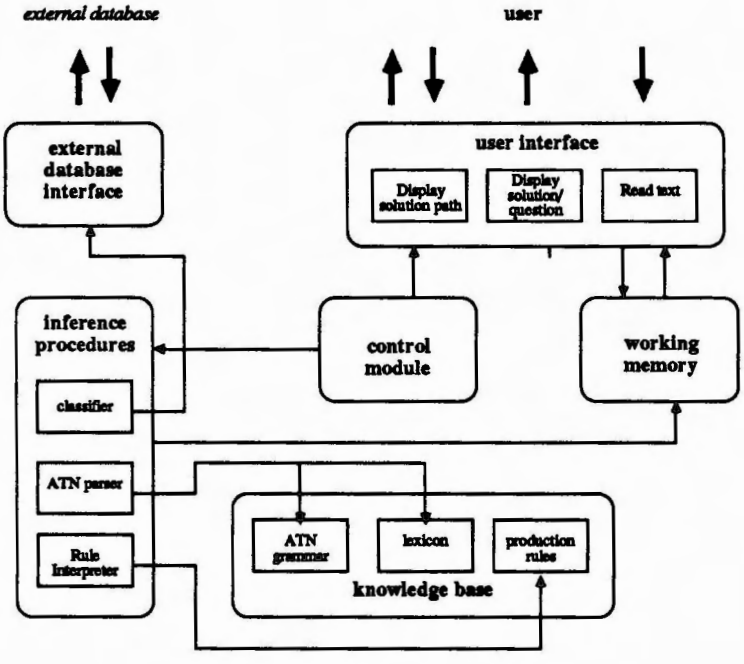


Figure 6: Architecture of the Flora Expert

There are three ways in which a paradigm can support the conversion of the analysis model into the design model. First, by advocating a particular functional block structure for representing the functions that need to be addressed by the final artifact. Second, by pointing out appropriate methods that can be used to achieve the behaviour needed to address particular functions. Third, by suggesting a particular physical block structure for aggregating the design elements that go along with the used methods. However, not all paradigms necessarily apply to the design process as a whole. A distinction can be made between *weaker* paradigms, which only apply to a particular part of the design process, and *stronger* paradigms, which apply to the overall design process. Obviously, strong paradigms will constrain the design process to a much greater extent than weak paradigms, i.e. the stronger the paradigm the more constraints it will put on the design decisions.

From the AI literature it appears that the notion of paradigms is often implicitly used. To illustrate this, consider the following two briefly described paradigms. A more extensive description of these paradigms can be found in [14]. A paradigm, that is used intensively, is the *production system* paradigm. This paradigm refers to the whole class of rule-based systems that are build in analogy with the traditional MYCIN system and its 'shell' version EMYCIN. Basicly, this paradigm supports at the top level two typical functions: consultation and explanation. The method most often used in the context of this paradigm is a backward search of an implicit AND/OR tree. It provides the overall behaviour that is needed to address the consultation and explanation function. The physical modules that are suggested by the production system paradigm are a knowledge base, an inference engine, a database, and an explainer, each containing a number of the design elements. The knowledge base, for instance, contains, among others, rules that are applied by the backward search method, whereas the inference engine contains the procedures, which are the physical parts of the backward search method that apply the rules.

An other example of a paradigm is the *state-space* paradigm. This paradigm is of particular interest because of the SOAR system [12] that is based on this paradigm. As mentioned before, we will not describe paradigms in detail. Here we will focus on the set of design elements and corresponding methods that are supported by the state-space paradigm.

Systems build with the state-space paradigm in mind, are characterised by explicitly representing the states in the problem solving process, consequently there is an initial state, a goal state, and a whole area of intermediate states. The initial state is transformed into intermediate states, in order to arrive at the goal state. Each states can, with help of heuristics, be analysed and classified as being more or less appropriate for achieving the final goal state. Typical for this paradigm is that the operators, which change states into other states, can as such not be classified as being good or bad. Finally, search procedures are needed to guide the search trough the state-space, i.e. to apply the available operators in a particular way. These search procedures are most typical for identifying the different methods. Some of those methods are: depth-first search, breadth-first search, hill-climbing, and others. More complex methods, within the context of this paradigm, are: GPS (General Problem Solver), A* (A-star), MiniMax, and others.

We are aware of fact that the descriptions presented here, have a tentative character. However, they clearly illustrate that a general framework can be identified which, on one hand, describes the general characteristics of many systems that have been build in the past, and on the other hand, provides support knowledge for systems to be build in the future. We therefore think it is essential that, like in the case of SOAR, more effort is put in formulating AI-paradigms. They could benefit the development of knowledge based systems to a large extent.

4 Conclusions

The use of these modelling tools have been tried out in various experiments. Because the modelling tools for the analysis of the expertise have been developed some 3 years ago, the amount of experience in this area is far more extensive than with the design models. From these experiences a number of conlusions can be drawn.

- The identification of elements of conceptual models in the data requires a good understanding of both the data and the modelling tools. In the KADS methodology this potential problem has been forestalled by developing so called "interpretation models" [5] which provide skeletal structures of conceptual models for generic tasks, like various types of diagnosis, assessment, prediction, planning, design, etc. A library of interpretation models is part of the KADS Power Tools in the KADS system, which support the development of the conceptual model for a particular domain.

  Although there is less experience with the design modelling language, it appears that the concepts themselves pose fewer problems to the KE than those of the conceptual model -if any. Therefore, there appears to be no need for readily available top-down structures. In fact, the initial problems in using the four layer language of the conceptual models may not be due to the vocabulary at all, but to the simple fact that the modelling of 'raw' data is always far more complex than the transformation, translation and adaptation of structures of a similar level of abstraction.

- The separation of modelling perspectives has proven to be very effective. If a single modelling perspective had been chosen, the knowledge engineer would have to be in two minds between data and implementation requirements. His task is 'naturaly' decomposed in a analysis activities of eliciting and analysing data on domain expertise, which are no longer confounded with external requirements, design issues and the actual implementation. Another benefit of this distinction is the fact that the process of knowledge engineering becomes more manageable, and that teamwork, based upon a specialised division of labour becomes possible. The documentation, and in particular the conceptual model has been a sufficient basis for a knowledge engineer to develop an adequate design model and implement the system with hardly any additional interaction with the 'conceptual modeller'.

- Finally, it is our hypothesis that the refinement, debugging and further maintenance of the KBS is facilitated by this approach, because the documentation not only provides well structured descriptions of the KBS, but the KADS system maintains links between data, models and specifications of requirements. This at least facilitates the understanding of the KBS if maintenance has to be performed by others than the developpers. Moreover, these descriptive tools may enable semi-automated refinement and debugging of a KBS. These issues are object of future research.

References

[1] A. Anjewierden. Knowledge acquisition tools. *AI Communications*, 0:29–38, 1987.

[2] S. Barthélemy, G. Edin, E. Toutain, and S. Becker. *Requirements Analysis in KBS Development,*. Esprit Project P1098, Deliverable D3 (task A2), Cap Sogeti Innovation, 1987.

[3] S. Barthélemy and N. Simonin. *Experiment F4: design and implementation.* Esprit Project P1098, Deliverable E4.2, Cap Sogeti Innovation, 1988.

[4] R.J. Brachman. On the epistemological status of semantic networks. In N.V. Findler, editor, *Associative Networks*, Academic Press, New York, 1979.

[5] J. Breuker, B. Wielinga, M. van Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, J-P Billault, M. Davoodi, and S. Hayward. *Model Driven Knowledge Acquisition: Interpretation Models.* Esprit Project P1098, Deliverable D1 (task A1), University of Amsterdam and STL Ltd,, 1987.

[6] B. Chandrasekaran. Towards a functional architecture for intelligence based on generic information processing tasks. In *Proceedings of the 10th IJCAI*, pages 1183–1192, Milano, 1987.

[7] W.J. Clancey. The epistemology of a rule based system -a framework for explanation. *Artificial Intelligence*, 20:215–251, 1983.

[8] W.J. Clancey. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.

[9] P. de Greef, G. Schreiber, and J. Wielemaker. *The StatCons Case Study.* Esprit Project P1098, Deliverable E2.3 (experiment F2), University of Amsterdam, 1987.

[10] G. Edin, P. Rooke, J. Breuker, T. Rangecroft, and F. Hayes. *The KADS Handbook.* Esprit Project P1098, Deliverable J1, Scicon Ltd., 1987.

[11] S.A. Hayward, B.J. Wielinga, and J.A. Breuker. Structured analysis of knowledge. *international Journal of Man-Machine Studies*, 26:487–498, 1987.

[12] J.E. Laird, A. Newell, and P.S. Rosenbloom. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[13] R. Neches, W.R. Swartout, and J.D. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Trans. Softw. Eng.*, 11:1337–1351, 1985.

[14] G. Schreiber, B. Bredeweg, M. Davoodi, and B. Wielinga. *Towards a Design Methodology for KBS.* Esprit Project P1098, Deliverable D8 (task B2), University of Amsterdam and STL Ltd,, 1987.

[15] B. Wielinga and J. Breuker. Models of expertise. In *Proceedings ECAI'86*, pages 306–318, 1986.

[16] E. Yourdon and L. Constantine. *Structured Design: A discipline of Computer Program and System Design.* Yourdon Press, 1978.

MACAO: A KNOWLEDGE ACQUISITION TOOL FOR EXPERTISE TRANSFER

Authors: Nathalie AUSSENAC, Jean FRONTIN, Jean-Luc SOUBIE

Address: Langages et systèmes Informatiques
Université Paul Sabatier
118, route de Narbonne
31062 Toulouse Cedex

Phone: 61.55.69.82

Keywords
Knowledge acquisition, psychological model

Abstract
Nowadays, the researches on cognition are characterized by its interdisciplinary nature. This short paper draws up the statement of a collaboration experienced between computer scientists and psychologists on the problem of expert's knowledge acquisition and modeling. Our contribution is explained in the context of existing knowledge acquisition literature. The MACAO project takes into account the specifications brought forward by a psychological model. It promotes particularly the expert's direct intervention and the analysis of his knowledge while he solves problems. Finally we present a first definition of a methodology supported by a computer program, as well as the problem arisen by the choice of a representation formalism.

1 Introduction

If we want the artificial intelligence softwares to really serve the purpose they have been assigned, we have to work seriously on the expert's performances the system can simulate. In an expert system, the knowledge base is the most difficult and time-consuming part. The completness and the relevance of the elicited knowledge from human experts are some of the major prerequisists for a efficient system with good performances. The M.A.C.A.O. project presented here has been performed in two steps. First, we have designed a methodology for knowledge elicitation. Then, in this methodology, we have introduced the use of an aid software to guide the expert and the knowledge engineer, and to make them play their part actively.

Until recently, researchers have tried to solve problems on knowledge representation rather than on the process of knowledge acquisition. This process was approximatively and empiricaly undertaken, only based on expert's interviews about the concerned area. However, recent researchers in Artificial Intelligence tend to define effective methodologies and tools to improve the expertise transfer and modeling. As cognitive psychology studies knowledge structuration and human reasoning patterns, it offers a set of techniques that might be of good help to give a better efficiency to the elicitation process. The numerous interests shared by artificial intelligence and cognitive psychology led us to collaborate with psychologists. The result of this co-operation is embodied in the integration of a psychological model in our project (see below). An expertise on the management of the bus network in Besançon, a French town, provided us the background for a validation.

In this paper, a short presentation on the state-of-the-art about the knowledge acquisition issue will be exposed in order to learly situate our study. The reasons for the co-operation with psychologists are stressed as an introduction to the presentation of the cognitive model of the expert. Then, the main specifications of the sofware are stressed before a detailed description of the methodology is presented. On each step, we will enhance the help the software may provide.

2 Why a psychological model ?

2.1 Studies on knowledge acquisition

As mentioned above, the acquisition of the expert's knowledge stands as a hindrance in the building process of a knowledge based system. To overcome this difficulty, many researches are carried out since Hayes-Roth & al's study that gave a set of guidelines about the process [HAYHES-ROTH,83]. Some focuss on defining more rational and efficient methodologies for the expertise transfer, whereas other ones aim at desinging proficient softwares as helps in order to reduce the bias produced by the knowledge engineer such as ETS [BOOSE,86], ROGET [BENNET,85], MORE [KAHN,85] or BERGSON [KONRAD,86]. Each of those tools proved to be effective in a determined area. But very few offer a solution to the whole problem [MAHE,87].

In fact, the knowledge acquisition is considered from two points of view in Artificial Intelligence [BOY,88], which have been balanced by Michalsky and Chilausky [MICHALSKI,80].

- On the one hand, expertise transfer is concerned. In this case, building a knowledge

base means retreiving human knowledge and encoding it to a computable formalism. Then, the main issues arisen are :

What kind of techniques make the expert elicitate his whole knowledge ?

What kind of knowledge representation prevents the knowledge from being distorted ?

- On the other hand, knowledge acquisition means learning process. In this second case, building a knowledge base consists in increasing a set of knowledge initially empty, and in providing devices that enable an automatic learning. New questions arise such as

  Which of the learning techniques is more easily computable ?

  What kind of knowledge representation suits better these techniques ?

  Teiresias by Davis [DAVIS,79], Agape by Ganascia [GANASCIA,87], Disciple by Kodratoff [KODRATOFF,85] or the software designed by Grumbach [GRUMBACH,87] are some of the interactive tools able to increase the knowledge in a base.

Even if knowledge acquisition is a quite new issue in AI, it has long been studied in cognitive psychology. The psychologists can provide answers. As far as learning is concerned, some results are quoted such as theories about generalization or problem-solving process [CAUZINILLE,85]. Moreover, cognitive psychology studies models of the expert and novice cognitive functioning. Those studies offer a double interest for AI . First, the models enable accurate modeling of the expertise. Their validation requires the use of techniques that can be integrated in knowledge retrieval and analysis methodologies [LOPEZ,86] [GAMMACK,84]. For instance, Boose builds ETS thanks to Kelly's personal constructs theory [BOOSE,87]. Otherwise, verbal protocoles might be significant ways of revealing a part of the expertise. Caverni and Le Ny have pointed out their advantages and some restrictions to their use [CAVERNI,86] [LE NY,85]. At last, Reitman and Reuter present a useful overview of the main psychological methods for knowledge acquisition [REITMAN,87]. They propose a classification according to their requirement to a knowledge engineer or not, and according to their suitability to the different kinds of knowledge.

Psychologists emphasize the suitability to different kinds of knowledge as a major concern. The complexity and the inaccessibility of this knowledge remain the main obstacles. This is what D. Berry aims at showing when she points out the need to retreive implicit knowledge however difficult it may be [BERRY,87].

Those works tend to prove that the use of such models may overcome the knowledge elicitation difficulty because it helps to enhance various aspects of the expert's behaviour. It also avoids the well-known tendency of the knowledge engineer who focusses only on the software representation pattern. The model reminds him to consider the whole knowledge, and not only the part needed to build up the base.

The influence of these statements can be notice in most of the studies in AI [GAINES,87]. Modeling the problem, the expert's reasoning process, the user's behavior is a new prerequisit for the design of methodology. But two different insights may guide those works. Several projects rely on the peculiar qualities of the knowledge representation in a software, such as ETS or MORE. Other studies aim at designing a method based on a formalism "knowledge

acquisition oriented". Their concern is to provide the expert with suitable techniques for an easy and painless transfer of his expertise thanks to the use of a cognitive model. Bennet opted this point of view when designing ROGET, as well as Hayward & al.

Hence, our purpose is to define a complete methodology based on the use of a tool to help the knowledge engineer, and on the integration of one of those psychological models of the expert. We feel more concerned by the latter point of view. So we first define the methodology choosing a knowledge representation in keeping with the model and the retreival techniques. This statements led us to design a software with the requirements specified in paragraph 4.

2.2 A description of the cognitive model

The model we use tries to study the structure of knowledge thanks to its evocation, storage and evolution [MICHEZ,86]. According to this, when an expert has to solve a problem, he refers to his knowledge by means of particular information selected in the environment: the objects of the expertise. These objects have physical characteristics and correspond to a mental model for the expert.

To depict the storage of the expert's knowledge, the model lies on different schemes and concepts.

1. Empirical schemes, located on a less abstract level, describe the knowledge used to solve and carry out some parts of problems: we call it a procedure. It consists of a set of instances and values. So solving a problem requires the call of several "procedures".

2. Conceptual schemes, located on an abstract level, describe the expert's knowledge on his strategies and on the choice of the convenient procedures: they are a kind of meta-knowledge. They gather conceptual relations and types of concepts.

This knowledge structured in two levels, when put together, form a set corresponding to the resolution of similar problems: we call them a "category of problem". The expert can describe those categories in natural language. Some particular objects of the higher level guide the expert to select the procedure to be applied when solving a given problem: we call them the determinants. All these concepts are the base for the analysis in the third part of the methodology.

In fact, we notice here some similarities with the output resulting of the analysis proposed by Hayward in [HAYWARD,87] : concepts and static links can be set on the domain level, whereas objects and dynamic links correspond to the inference level. We also find some analogy between the "procedures" and the instanciations of the meta-classes of problem defined by Hayward & al. At last, what they define as the class level could involve the "categories of problems".

3 The chosen expertise

The expertise we use to validate our results consists in overseeing eight bus-lines in Besançon. Each bus is connected to a central computer which records information about the evolution of the network situation. An operator may display the line he wishes on the

screen. He is given information such as the position of each bus on the line, bus deviation relating to the scheduled time... If he detects a disturbance (bus in advance, late, "bus-train" for instance), he gives some instructions to the concerned bus driver in order to correct the disturbance. He might ask him to stop the bus for example. Meanwhile, alarms can be displayed. They may concern a bus which is not on the screen. An alarm points out a particular disturbance of great amplitude. There are three possible kinds of alarm: "advance-before-town-center", "retarded-departure", "slowness-at-next-departure".

This expertise was studied by psychologists during several months [MARINE,88]. Four persons have been observed: two experts and two novices. From all these observations, the psychologists have elicited each subject's knowledge and behaviour. We have all the elements to build up a knowledge base which would simulate an expert or a novice. This experiment provides means to control the accuracy of the formalism -a graph with objects and links- , and its ability to represent the whole expertise. An other kind of experiment has to be carried out to validate the protocole analysis step.

This kind of expertise is quite similar to a diagnosis expertise. But it is slightly different because, while solving a given problem, the subject is bound to face a new one - an alarm may be detected or anything else -. In fact, the main difference is that the situation is no longer a static one, but changes. This underlines the importance of the dynamic links in the graph: they have to express those changes.

Then, it might be interesting to apply this methodology to other expertises concerning the same range of problems. This could help to define more precisely the different kinds of links and their management.

4 The specifications of the acquisition tool

Before the MACAO methodology is described in detail, some specifications of the software are introduced.

This software has been designed respecting the following fundamental properties:

- independance from the knowledge process
- suitability to the knowledge engineer as well as to the expert
- keeping of the whole meaning of the expert's speech and procedures by mapping the linguistic data to knowledge.

For these purposes, as a key to this suitability, an expert's psychological model is to be taken into account thanks to adapted acquisition techniques. The representation of the knowledge has also to be adaptable enough to be as reliable as possible to the expertise, rather than computable.

4.1 General requirements

The acquisition tool is meant both for the expert and the knowledge engineer. Thus, the expert is invited to take a greater and more active part in the process. It is a way to make him feel more concerned by the expert system building. In fact, the software stands as a memory of the expertise which makes the analysis by the knowledge engineer easier.

A special attention has been paid to the man-machine interface. In fact, an expert doesn't naturally speak a specific formalism, such as "production rules" for instance. Moreover, we don't want to impoverish his speech and more generally his knowledge by constraining him to articulate his expertise in an unfamiliar environment. He "transfers" his knowledge mainly through words representing the objects he handles (functional units such as concepts, characteristics or values ...), and their relationships. He is free to create as many items as he needs. To simplify his dialog with the machine, he is offered the use of convenient tools with a mouse, icons and graphics. We believe that a graphic presentation of the knowledge structure is a powerful device for the expert to control the formalized knowledge on each step.

4.2 The knowledge representation

At last, we would like to stress a few points about this formalism. We assume that it will be all the more convenient and efficient as it satisfies the following properties:

- ability to represent any kind of knowledge (declarative as well as procedural), with an important semantic capacity;
- readability and independance of the representation in the later knowledge base;
- balance between an easy management and a powerful computability;
- closeness the language to facilitate communication with the expert.

This choice also raises the question whether the formalism has to include the mechanism that processes the knowledge or not. This means that the logical level to be refered to has to be selected. As we want to give priority to the semantic ability, the chosen formalism focusses on objects. It only alludes to reasoning without being able to carry out inferences.

The knowledge is represented by two kinds of diagrams: graphs on the expertise level , where the nodes are the objects connected by links of different kinds, and graphs on the conceptual level ones, where the nodes are concepts. The graphs of the lower level can be considered as instanciations of the conceptual ones. This hierarchy of diagrams turns the final knowledge representation into a superposition of graphs. To homogenize this structuration, the nodes will be called "functional units" in both graphs. those units are the actual items handled by the expert and the knowledge engineer. This representation facilitates the adaptation to the user's dialog and explanation levels in the man-machine interface of the further knowledge-based system. Furthermore, it provides a convenient tool for the expert's direct control of the knowledge consistency.

In both kinds of graphs, we separate two general classes of links, according to their meaning:

1. static links describe the world of the expertise (they connect an object to its different attributes: characteristics, properties or values);
2. dynamic links describe the resolution process followed by the expert (a given object is calculated from, depends on, ... the ones it is linked to).

If the expert can express precisely most of the links without ambiguity, some of them remain uncertain. He might "feel" that two objects are in relation, but not be able to define in which way (for instance, two objects seem to have an analogy). So we distinguish the first links, called "strong links" from the second, called "weak links". A tool-box can help the expert and the knowledge engineer to handle the links. It provides a list of labels for the links as well as a set of instructions to link or un-link functional units.

Therefore, the acquisition methodology can be run whatever the later expert system may be. The MACAO software is actually to be considered as a separate module in the system.

5 Presentation of the methodology

The methodology is based on the observation of the expert at work. The analogies with Breuker and Wielinga's model previouly emphasized led us to compare the following stages to the level of knowledge analysis mentioned by Wielinga in [WIELINGA,84].

The first stage is a dialog between the expert and the knowledge engineer. They specify the user's environment, the categories of problem to be solved and the main features of the expertise. This set of information stands as the software initialisation nucleus.

The second stage is carried out with the prospect of a protocole analysis. The expert is invited to solve several problems of the different categories he has mentioned. He is placed in a situation as close as possible to reality. Meanwhile, the knowledge engineer asks him for additional information. The expert's explanations are recorded. Both stages correspond to Wielinga's knowledge identification level.

During the next stage, the knowledge engineer analyses the information retreived through the frame suggested by the model -similarities and differences concerning the resolution of each problem-. To make this work easier, the different knowledge graphs can be displayed. This analysis gathers the knowledge conceptualization and the epistemological analysis defined by Wielinga.

As the expert is the only one to judge the accuracy of the retreived and structured knowledge , he validates it in the last stage. He may correct the conceptual graph or even the ones corresponding to the glossaries. We suggest him to solve problems derived from the one proposed in the second step, and to compare the items of the path in the graph with the information he uses.

With the help of the engineer and of such a tool, the expert may present the whole knowledge of his expertise. At the end, the program has memorized a final data structure which describes the expertise session in a formalism independant of the later knowledge base.

Now, we shall present more precisely the four stages of the methodology.

5.1 The expertise identification

At the beginning of the process, the expert and the knowledge engineer meet several times in order to define a general environment. After those interviews, the expert feels more concerned by the project and the knowledge engineer has become acquainted with the task area. He knows the expert's general behaviour. From the examples quoted by the expert,

he deduces the different "categories of problems" dealt with, and the main objects handled. Therefore, the speech analysis aims at defining the expert's ability, at structuring roughly his knowledge, and identifying his basic reasoning. The analysis of the objects helps the engineer to build up a lexicon which gathers the concepts characteristic to solve the problem. It is the starting point of a more systematic retrieval with the aid of the computer tool.

The software is then initialized with data that specify the different "categories of problems" to be solved by the later system (the associated aims to be reached, the basic hypothesis and some of the objects) and with questions corresponding to several problems of these categories. Each of them refers to a genuine problem the expert has already faced. They could be some of the later user.

The program suggests the expert to link some objects to one of the categories. All the aims to be achieved can be explicit in the lexicon.

Furthermore, the elicited knowledge consists of the different "categories of problems" to be overcome, needed to solve the problem .

5.2 Use of the expertise

During this experimental step, the lexicon is to be expanded. The relationships between the objects and the corresponding graphs are specified. It enables the elicitation of the information about the knowledge structure and its use.

To this end, the expert is asked several questions selected in keeping with his explanations during the former interviews and the categories of problem they are related to. This simulation is important because the selected model states that expert's knowledge is acquired through real experiences and is fully used only when the expert has to overcome real problems. Thus, it seems natural to retrieve the expert's knowledge from his solving a real case. At first, it is more convenient to solve easy problems in order to surround closely the main functional units.

For each question, the expert simulates his expertise and runs the case. He specifies it, step by step, shows the informations he needs, the objects he uses. He can tell their value, the range of value expected, or the set of values corresponding to this case. He can draw links among those objects. As for the links, a tool-box provides a set of commands to add, select or suppress objects. In fact, the expert draws up, closely and methodologically, an inventory of the elements he needs. The program memorizes these data by sequentially recording them in one file per problem. Such a file is called the "glossary" of a given problem.

Meanwhile, the knowledge engineer urges the expert to think aloud, to comment his behaviour throughout the solving process. This simultaneous verbalisation is recorded. The expert is asked to answer the following points:

- some reasons for the need of an information at the moment it is alluded to;
- the means to reach it or to modify it;
- its consequences upon the problem or other entities;

5.3 Analysis of the knowledge structure

The third stage is a systematic analysis with a double aim. On the one hand, it helps detecting coherence defaults, redudancies and incompletness. On the other hand, the knowledge engineer completes the lexicon and chiefly structure the knowledge using the recorded verbalisations.

The information analysed is of two kinds: static (knowledge structure) and dynamic (inference strategies). An instance of a static relationship is : a bus "is-characterized-by" its number, the deviation relative to the scheduled time, ... An example of a dynamic relationship is the one meaning that alarms are delt with only when all the disturbances are overcome. This information has two sources: the glossaries and the recorded expert's speech.

First, the engineer clusters the objects from the glossaries and defines the concepts which they may refer to. He connects those concepts with links similar to those existing between the corresponding objects. These concepts and the static links among them form the basic static knowledge.

Then he considers the knowledge elicited within a category of problems. By watching the differences from one glossary to an other and the influence of those variations, he puts forward specific functional units. The proper reasoning process and "procedure" in this category are defined by the common sequences of objects in the glossaries.

The comparison of the glossaries from the different categories provides information such as the "determinants" which characterize each category of problems. The similarities between all the glossaries represent the expertise knowledge.

5.4 Approval and validation of the knowledge by the expert

Links may be modified by the expert: the objects of an expertise are generally easily known but the links may often be changed during the expertise description: the expert cannot always express most of the links precisely and without ambiguity. Let's suppose he tells a sentence in natural language to define a relationship between two objects: it will be necessary to study the semantics expressed. An analogy with one of the existing relationships or links may be found. So the expert is told to express this relationship with a particular link. If he is not sure enough, the link will be a "weak" one. But the actual aim is to reduce these links to strong ones.

We shall distinguish two steps in this last stage. First, the expert has to control and refine the conceptual graphs built by the engineer. He may also modify the graphs on the lower level. He improves the structured knowledge until he is satisfied with it. Uncertainty can be expressed by adding new links.

In a second time, the purpose is to test and validate this set of knowledge, as well as to achieve completness and concistance. The expert is suggested to solve particular problems derived from the cases he previously ran. The only changed units are the values of the objects identified as "determinants". For each problem, he compares the knowledge graph he has built up with the conceptual one. Following an iterative process, the graphs are updated for a better suitability to his expertise.

6 Conclusion

Up to now, the knowledge acquisition systems were dedicated to the design of knowledge bases running with specific inference engines. We need today an expertise gathering more complete and nearer of the expert cognitive reality, by means of psychological model. We need as well a complex data structure which will provide the access to kinds of knowledge not easily workable through classical and non-classical logic, moreover wich may help the final user only by displaying it at right moment.

The first investigation field we have chosen for MACAO is related to the knowledge of temporal and spacial events. The general purpose conclusions yielded by this experiment will increase the amount of predefined links in the acquisition system. It is necessary that such a tool should be evolutive to take into account new relationships between different concepts in other expertise areas.

References

[BENNET,85] BENNET J. S., A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system. *Journal of Automated Reasoning*, 1, 49-74. 1985.

[BERRY,87] BERRY D., The problem of implice knowledge. *Expert Systems (4)*, 3, 114-151. August 1987.

[BONNET,86] BONNET A., HATON J. P., TRUONG NOC J. M.Systèmes experts : vers la maîtrise technique. *Interéditions*, 288 pages. 1986.

[BOOSE,85] BOOSE J. H., A Knowledge Acquisition Program for Expert System Based On Personal Construct Psychology. *International Journal of Man-Machine Studies (23)*.1985.

[BOOSE,86] BOOSE J. H., Expertise Transfer for Expert System Design. *Elsevier Ed., "Advances in Human Factors/Ergonomics"*, 3, 312 pages. 1986

[BOOSE,87] BOOSE J. H., BRADSHAW J. M., Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. *Int. Journal of Man Machine Studies (26)*, 3-28. 1987.

[BOY,88] BOY G., FALLER B., SALLANTIN J., Acquisition et ratification des connaissances. *Actes des 2emes journées Internationales du PRC-GRECO, Intelligence Artificielle*, TouLouse, 321-356, mars 1988.

[CAUZINILLE,85] CAUZINILLE-MARMECHE E., MATHIEU J., Processus cognitifs d'acquisition de connaissances. *COGNITIVA*, Paris, 1985.

[CAVERNI,86] CAVERNI J. P., Les protocoles verbaux comme observable des processus cognitifs. *Colloque Annuel SFP, " Activités cognitives : modèles de processus et niveau d'observation "*, Aix-en-Provence, mars 1986.

[DAVIS,79] DAVIS R., Interactive transfer of expertise: acquisition of new inference rules. *Artificial intelligence (12)*, n 2, 121-157, 1979.

[GAINES,87] GAINES B., R., An overview of knowledge-acquisition and transfer. *Int. Journal of Man Machine Studies (26)*, 453-472. 1987.

[GAMMACK,84] GAMMACK J., YOUNG R., Psychological techniques for eliciting expert knowledge Research and Development in Expert Systems. *4th Technical Society Specialist Group on Expert Systems*, Warwick, 105-112. dec. 1984.

[GANASCIA,87] GANASCIA J., G., AGAPE : de l'appariement structurel à l'apprentissage. *Intellectica : Apprentissage et machine (1)*, n 2/3, 1987.

[GRUMBACH,87] GRUMBACH A., Contribution à l'étude de modéles d'apprentissage en interaction avec un environnement inspirés du comportement humain. *Thèse de 3ème cycle, ENST*, Paris, 1987.

[HAYHES-ROTH,83] HAYES-ROTH F., WATERMAN D., A., LENAT D., B., Eds,Building Expert Systems.*Addisson Wesley*, Reading: Massachussets, 1983.

[HAYWARD,87] HAYWARD S. A., Wielinga B. J., Breuker J., A., Structured Analysis of Knowledge. *Int. Journal of Man-Machine Studies (26)*, 487-498, 1987.

[KAHN,85] KAHN G., NOWLAN S., MAC DERMOTT J., MORE: an intelligent knowledge acquisition tool. *Proceedings of Nineth International Conference on Artificial Intelligence*, Los Angeles, California, 637-639. 1985.

[KAHN,85a] KAHN G., NOWLAN S., MAC DERMOTT J., Strategies for Knowledge Acquisition. *IEEE Transactions on pattern analysis and machine intelligence*, vol. PAMI-7, NO. 5, sept. 1985.

[KLINKER,87] KLINGER G., BENTOLILA J., GENETET S., GRIMES M., MAC DERMOTT J., KNACK - Report-driven knowledge acquisition. *Int. J. of Man-Machine Studies (26)* 65-79, 1987.

[KODRATOFF,85] KODRATOFF Y., Une theorie et une méthodologie de l'apprentissage symbolique. *COGNITIVA 85*, Paris, 1985.

[KONRAD,86] KONRAD W., BERGSON: a language and environment for methodological knowledge acquisition as evolutionary problem modeling out of epistemic primitives. *Proceedings of International Join Conference on Artificial Intelligence*, Munich, Germany. 1986.

[LE NY,85] LENY J., F., Compréhension du langage et acquisition de connaissances. *COGNITIVA*, Paris, 1985.

[LOPEZ,86] LOPEZ DE MANTARAS R., AGUSTI J., PLAZA E., Knowledge elicitation and analysis for approximate reasoning systems. *CIIAM 86, Intelligence Artificielle*, 91-107, 1986.

[MAHE,87] MAHE H., VESOUL P., Acquisition des connaissances et adaptation à l'utilisateur: outils et méthodes. *Avignon 87, 7ème journées de l'I.A.*, 625-645, 1987.

[MARINE,88] MARINE C., CELLIER J., M., Méthodologie d'extraction des connaissances: comparaison de protocoles verbaux débutants experts. *ERGO'IA 88, Afcet*, Biarritz, 1988, (to be published).

[MICHALSKI,80] MICHALSKY R., S., CHILAUSKY R., L., Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. *Int. Journal of Man-Machine Studies (12)*, 63-87, 1980.

[MICHEZ,86] MICHEZ B., Approche du fonctionnement cognitif: schéma empirique et schéma conceptuel dans le processus de structuration des connaissances. *Mémoire de DEA Université de Toulouse le Mirail.* 1986.

[REITMAN,87] REITMAN OLSON J., REUTER H., Extracting expertise from experts: Methods for knowledge acquisition. *Expert Systems (4)*, n 3, 152-167, August 1987.

[DE SAINTE-MARIE,87] DE SAINTE-MARIE C., Comment MACHIN devint chamelier. *Intellectica (1)*, n 2/3, 70-99, 1987.

[SHACHTER,87] SHACHTER R., HECKERMAN D., Thinking backward for Knowledge Acquisition. *AI Magazine (8)*, n 3, 55-61, Fall 1987.

[WIELINGA,84] WIELINGA B., J., BREUKER J., A., Interpretation of verbal data for knowledge acquisition, *ECAI 84, Advances in Artificial Intelligence*, T. O'Shea Ed., 41-50, 1984.

Barbara Becker

Institute for Applied Information Technology
German Research Institute for Mathematics and Data Processing
Schloß Birlinghoven
Postfach 1240
D-5205 Sankt Augustin 1

Towards a concept for case-based knowledge-acquisition

0 Abstract

The aim of this article is first to draw attention to some peculiarities of human expertise.which are reason for many difficulties in knowledge acquisition. This is realized by an analysis of human expertise, for which I will draw upon results from cognitive psychology . Then, I will discuss whether a case-based approach offers an alternative to traditional methods in knowledge acquisition . Finally, I draw up a first model of case-oriented knowledge acquisition which may take into account some general characteristics of human expertise which until now defy a precise description by the experts .

1 Introductory remarks

Looking at the current literature on knowledge acquisition, it becomes apparent that there is a fundamental theoretical background missing here. On thinking about possible approaches towards a theory on knowledge acquisition, there are in general two possibilities:

- a theoretical access, i.e. to look for a meta-scientific basis, which may lead to a consistent methodology,
- a pragmatic approach, that is to analyse the peculiarities of human expertise and try to find adequate methods which take into account these specific attributes.

Considering the current situation in knowledge acquisition it seems to be fruitful to look at human expertise in more detail, because the methods of acquiring expert knowledge must take into account the specific kind of knowledge structures which seem to be typical for experts. An analysis of human expertise may lead to new concepts and methods in knowledge acquisition or to a higher priority of traditional approaches. Besides, investigations on human expertise provide a basis to evaluate the choosen methods.

2 Characteristics of human expertise

During recent years, in the context of expert system development, several investigations have been made into the organization of expert knowledge and into problem solving processes in the field of cognitive psychology. Even if these surveys elucidate only a small part of the expert´s competence, it might be productive to take into account what seems to be characteristic for human expertise.

Some of the most significant results of such investigations have come out of the comparison of novices in a particular field with recognised experts (for example: Kolodner 1983, Benner 1984, Larkin 1980, Gruber/Strube 1987). Chi, Feltovich et al. (1982) discovered that the expert is distinguished from the novice by both a collection of domain specific heuristics and by a complex

network of information and a structuring of the domain specific knowledge adequate to the problem.

The manifold association of concepts is as important for effective problem solving as the clustering of information, which depends on subjective experience and which can be modified according to new findings. Experts tend to make use of their knowledge by not only considering the dominant facts of an actual situation but also by taking into account aspects which seem to be unimportant at first glance but which prove to be relevant afterwards.

Expertise is a complex and dynamic process . So the knowledge of an expert does not correspond to a specific optimal knowledge status, but is subject to continuous change caused by the wealth of his/her subjective experience of every day. This experience of experts, which leads to their competent behaviour even in difficult situations, is based on the fact that they have handled a lot of problem-situations and domain-specific cases. These cases form a background for the concrete problem solving behaviour by offering the possibility to find analogies between former situations and the actual one. Experts tend towards building collections of comparable cases which enable them to find rather quickly an adequate strategy to solve the problem at hand.

Furthermore, the characteristic ability of experts to deal with incomplete and uncertain information and to generate hypotheses even on the basis of vague data, is based not only on theoretical knowledge of a domain but on the personal experience which has been gained during a long phase of practice. It has been shown in cognitive psychology that it is rather difficult to handle incomplete facts only by using theoretical knowledge and that it is quite more effective to refer to analogous situations to find a possible way to act or find a correct decision even on the basis of such uncertainty.

Another important characteristic of human expertise is the capacity to revise solution hypotheses or the steps taken because of them, whenever inconsistent data comes up. Novices often tend to overlook contradictory facts and very often misjudge the relevance of contradictory data to the generated hypothesis. The adaequate handling of such information and the capability to rate the importance of new facts are major factors in the experience of an expert

Furthermore, experts are able to judge trends and to take tendencies into account in their decision making processes. During the course of their problem-solving, experts have to continuously reassess the potential results of their chosen strategies; in doing so they go back both to their own subjective experience and to the objective knowledge of the domain.

Another important peculiarity of human expertise is the capability to alter between different forms of mental representation. The so-called modification of viewpoint regarding a case depends on the potential availability of different knowledge structures. The transformation of one representation into another, for example a linguistic description can be less efficient than a pictorial representation of a situation or an object, seems to be an important point in effective problem solving. Furthermore, often it is necessary to enlarge the point of view by taking into account side-effects and implicit conditions of an actual situation - novices very seldom have this capability and tend to be limited to one representation. Usually, experts are capable of constructing different mental representations of a problem, for example, they are able to make analogical and propositional, complex and simple, abstract and detailed, subjective and objective representations of a problem situation and may use any of these at their discretion. As early psychological investigations on the mental structuring of a problem space already have shown (Wertheimer 1945), people are often very successful in dealing with complex situations just because they are capable of restructuring their internal representation of the problem at hand whenever this is necessary.

At least, another characteristic attribute of experts is their flexibility and adaptability in unknown situations. Experts do not only make use of their large domain knowledge, but refer to their non-expert knowledge, too. It is very often this everyday knowledge which enables experts to regard a situation from an unusual point of view. This "common sense", as it is called, offers the chance to find innovative strategies, because the apparent departure from known ways and domain specific strategies may lead to creative ideas. The relevance of common sense for domain specific problem solving processes is disclosed for example in medical decision making; Roessler and Viefhues pointed out, that the physician is only capable of treating a patient in an adequate way, because he/she lives in meaningful situations; he/she collect continuous experience during the concrete interaction with his/her environment which enables him/her to recognize the specific meaning of a patient´s illness.

To sum up, investigations of human expertise made apparent the complexitiy of the expert´s knowledge and suggest that the actual approach in knowledge acquisition has to be enlarged by new methods under consideration of these peculiarities.

3 General reflections about human expertise

On regarding human expertise in general, it becomes apparent that the expert´s knowledge most essentially depends on the current situation in which it is used. In contrast to a rationalistic view, which implies a deterministic separation of subject and object, Winograd/Flores (Winograd/Flores1986) and Lischka (Lischka1987) argue with reference to Heidegger, that an expert is not outside but always within a given situation. The concrete problem situation, in which the expert has to decide and to act in an adequate way, interacts with the expert´s knowledge and influence his/her kind of decision in a fundamental manner. So expertise is directly associated with the context in which the expert has to use his/her knowledge. In each decision-making process, the expert and the situation form a specific relationship. Efforts to detach the subjective expertise from the context where this knowledge is being applied- as is, after all, the precondition of knowledge acquisition - necessarily leads to a reduction and to a wrong impression of the expert's knowledge.

Furthermore, human expertise cannot be separated from the interactional context in which the knowledge first arises, becomes effective and is then continuously modified. Experts are not isolated beings, but live in a social context, in a society, which influence them fundamentally. Their expertise is not static, remaining unaltered once it has been established at a level designated as expert. Rather, expertise is the result of a permanent exchange of information and ideas between experts and non-experts. Discussions of strategies and controversial views with other experts, and especially the fact that the verbalization of subjective ideas and the justification of personal opinions in the presence of colleagues leads to an increasing clarity and definition of their own thought; all th is is a fundamental precondition of human expertise.

Looking back to the psychological investigations about human expertise, it becomes apparent that the subjective experience

plays an important role in the expert's competence. This experience has been gained in the concrete, everyday practice by solving problems and dealing with specific cases. In doing so, experts attain a conclomerate of private knowledge which differs from the objective theoretical knowledge about a domain. Experts not only learn to pursue skills but they also have a rather clear recollection to a lot of cases and problem situations which they have met during their professional activity. Both, the practical know-how and the wealth of experience create the personal competence which distinguishes one expert from the other and which goes far beyond the ad hoc describable objective knowledge of a specific field.

In this context, it might be relevant to look at the difference between experience and rationality. Experts are endowed with highly differentiated factual knowledge and complex problem solving strategies, but their specific competence is fundamentally based on the experience which they have gained by handling concrete cases. The direct interconnection of practical experience and theoretical abstraction becomes apparent in the specific ability which characterizes an expert. It forms the individual competence which is more than either one of these two components, because expertise cannot be solely put down to concrete experience on the one hand or to rationality on the other. Both the complex perception of reality, as it is always immanent in practical experience, and abstract reflection, as it characterizes theory, form a complementary framework, which is an essential attribute of expertise (Becker 1987). As long as there exists no theory of practice, there are limitations to the acquisition of expertise, as it is directly connected with practical experience; only the rational dimensions of knowledge are accessible, but these are devoid of the specific expression which would be lent to them by the perception, knowledge and actions of the subject . Knowledge acquisition- especially when it takes place on a general level (see for example in KRITON, Diederich 1987)- is restricted to determining practice only in the form of abstract descriptions of skills, strategies and situational behaviour; it is clear that this acquired knowledge is no longer comparable to the actual, subject-specific expertise of an individual expert.

The separation of knowledge from the concrete experience of the knowing subject, as it usually takes place in knowledge acquisition, leads to a reduction of knowledge, if the limits of traditional methods of knowledge acquisition process are overlooked. But even, if not possibly forever there will remain some indescribable dimensions of human expertise which exceed

from verbal description or observation they may exist some methods in elucidating human knowledge which are more adequate to take into account some of the above mentioned pecularities of human expertise.

5 Towards a case-oriented approach in knowledge acquisition

As we have seen, one of the most important characteristics of human expertise is the subjective experience of an expert which he/she gains during his/her professional activity. Among others, experience is the capability to go back to a lot of cases which have been solved by the expert in the past. Investigations on human expertise suggest that the expert´s knowledge is - to a large extent - created by such a case-specific competence. It has been shown very often, that the meaning of concepts and the judgement about the adequateness of strategies is not separable from the context in which the knowledge is used. In problem solving situations this context is most essentially determined by the actual case and the expert´s knowledge of how to handle it. If we aim at considering the peculiarities of human expertise we both have to refer to the context-dependency of human knowledge and the situational involvement of the expert while dealing with an actual case. The context as well as the situation are formed most essentially by the current case. Therefore, it seems to be an adequate way to enforce a case-oriented approach in knowledge acquisition.

The following ideas are still rather vague and indistinct - they shall yield a general case-based concept for knowledge acquisition as it has already been described to a limited extent by Chandrasekaran et al. (1983), Kolodner (1987) and Bartsch-Spörl (1987).

To illustrate my thoughts I will configurate a fictive knowledge acquisition plan which is based on the available methods of KRTION, a knowledge acquisition tool which has ben developed by Diederich et al. (1987). This approach results from my practical experience in cognitive psychology, especially from interviewing experts in medicine to elucidate their case-specific knowledge and may most generally be described as a case-based concept for knowledge acquisition.

In the beginnig of such a procedure a prototypical case should be identified in cooperation with experts of the domain. (In medicine, this might be the case of a patient who shows a lot of typical

attributes and symptoms of an illness). After selecting such a case, a recognised expert shall describe the general strategy which he/she will use to solve the problem at hand. If it is possible, this should lead to a global problem solving frame, which might be illustrated by a graph, a decision tree or a more complex graphical illustration of the general procedure and which will function as a ruster for the further course of action.

By identifying this general case-specific frame, a lot of concepts and attributes will be mentioned by the expert. Therefore, in the second step, each part of the global ruster should be analysed in more detail. The most important concepts should be identified and described by refering to both traditional methods of knowledge acquisition as there are the repertory grid technique and the laddering technique, and methods of cognitive psychology, for example the category research of Rosch et al. (1978). Especially the category research offers a very interesting approach for identifying the definitive and the characteristic attributes of a concept according to the actual case. This may be illustrated by an example:

The category of an illness includes one or several so-called "leading" symptoms, which are the definitive attributes of the specific category and wich function as an important evidence for the appropriateness of the category in a specific case. Besides, there are several characteristic attributes of a category which may confirm or weaken the selection of a category. Research in cognitive psychology (Rosch/Mervis et al., 1975) has shown, that the orientation in a new situation - and these are problem solving processes, too - often aim at finding an adequate category by looking for its definitive and characteristic attributes. and matching them with the data of an actual case. In a case-based knowledge acquisition approach, experts should be motivated to identify the definitive and the characteristic attributes of each concept which has been mentioned during the description of the general problem solving frame.

After identifying the most important concepts and their attributes, the problem solving frame should be regarded in more detail. Isolated parts of the global frame, for example subgoals or single decision steps, should be analysed by protocol analysis to elucidate the procedural and strategic knowledge which is necessary to handle the given case and to lead from one problem state to another. Even if such an approach at least depends on the problem-solving concept of Newell and Simon (1972) which has to be regarded with a sceptical view, this concept offers a possibiliy

to structure the elucidation process and to facilitate the verbalization of important knowledge so far as this is describable.

The continuous interaction of category- and protocol analysis, should bring up a "filled" frame structure which characterises the problem solving behaviour of an expert in a specific case and takes into account the context dependencies of the semantics of categories and procedures. The interactional consideration of both declarative and procedural knowledge will take into account that these both kinds of knowledge cannot be separated from each other but are connected very closely.

After having filled up the primary frame of a specific case, other cases should be elucidated in the same manner, so that at least we will have a case-library. It would be ideal, if clusters of cases can be identified, which include analogies and lead to a case-oriented structuring of the expert´s knowledge. First efforts in this direction have been made by Tversky (1977) to identify similarities between cases which may lead to a more structered representation of cases in a knowledge base.

The case-based approach in knowledge acquisition implies the advantage to refer to a "natural" way of knowledge representation, because experts very often find an access to their individual knowledge by finding analogies between an actual problem and some already solved cases. Therefore, the case-library, which will be the result of such an approach, may lead to case-based resoning (Kolodner 1987) in a knowledge-based system, in which cases function as the basis for inference processes in a problem solving situation, because they may be compared with a kind of memory for past experience. The methods for generalizing case knowledge have to be explored and elaborated - we aim at doing so in the context of this project.

To sum up, the case-oriented approach aims at a more structured and context-dependant way of acquiring knowledge. Furthermore, I believe that the solution of a specific case may elucidate some implicit knowledge structures which remain hidden by a general description of the domain-specific knowledge.

6 Concluding remarks

Irrespective of what new methods or modifications to traditional strategies of knowledge acquisition to suit requirements may be found, the question to be discussed always is whether there are fundamental limits of acquiring expertise, which result from the ontological peculiarity of human knowledge. The dynamic and provisory character of expertise and its openness seem to be such attributes.

This indefiniteness of expertise should not be overlooked in view of knowledge acquisition, because it fundamentally characterizes human experience in general. The separation of knowledge from the concrete experience of the knowing subject, as it usually takes place in knowledge acquisition, leads to a reduction of knowledge, if the limits of the acquisition process are overlooked. Not only will expertise be reduced to its "rational part", which is only one dimension of it, but scientific consequences are also conceivable, which could lead to a successive elimination of some special kinds of thinking, which are inevitably linked to human existence.

The hermeneutic, meaningful action in the real world, presupposes human practice, beause only here does man experience, that he lives in a meaningful context.
He is able to perceive and understand his environment by being situated within a world and within a tradition. The separation of human knowledge from the practice which creates this knowledge and is fundamental to every skill, implies the danger of an objectivation of all thinking - subjective expertise will become increasingly unimportant whereas the standardized knowledge continuously begins to take its place.

Bibliography

Chi, Feltovich et al.Chi, M.Th., Feltovich, P.J., Glaser, R., "Categorization and representation of physics problems by experts and novices", in: Cognitive Science 5, 1982

Schwartz, S., /Griffith, T., "Medical Thinking", New York, Berlin, Heidelberg 1986

Roessler, W., Viefhues, H., "Medizinische Soziologie", Stuttgart 1978

Dörner, D. et al., "Lohhausen. Vom Umgang mit Unbestimmtheit und Komplexität", Bern Stuttgart Wien 1983

Wertheimer, M., "Productive Thinking", New York 1945

Becker, B., "Wissen und Können", in: Morik (Hrsg), "GWAI-87", Informatik Fachberichte, Springer 1987

Winograd, T., Flores, F., "Understanding Computers and Cognition", Norwood, N.J., 1986

Rosch, E., Mervis, C.B., "Familiy Resemblance. Studies in the internal structure of categories", in: Cognitive Psychology, No. 7, 1975

Rosch, E., "Principles of categorization", in: Rosch, E., Lloyd, B. (Eds), "Cognition and Categorization", Hillsdale, N.J. 1978

Diederich, J., "Wissensakquisition", GMD-Arbeitspapier No. 245, St. Augustin 1987

Lischka, Ch., "Über die Blindheit des Wissensingenieurs, die Geworfenheit kognitiver Systeme und anderes...", in: KI 4 / 87, Oldenbourg Verlag

Bartsch-Spörl, B., "Behandlung von fallorientiertem Erfahrungswissen in Expertensystemen", KI 4 / 87, Oldenbourg-Verlag

Kolodner, J.L., "Toward an understanding of the role of experience in the evolution from novice to expert", in: International Journal of Man-Machine Studies, 19, 1983

Larkin, J., et al., "Expert and novice performance in solving physics problems", in: Science 208, 1980

Benner, P., "From Novice to Expert", Addison-Wesley 1984

Gruber, H., Strube, G., "Zweierlei Experten: Problemisten, Partiespieler und Novizen beim Lösen von Schachproblemen", Max Planck Institut, München 1987

Kolodner, J., "Extending Problem Solver Capabilities through Case-based Inference", in: Langley, P. (ed), Proceedings of the 4th Int. Workshop on Machine Learning, Irving 1987

Tversky, A., "Features of Similarity", Psychological Review, Vol. 84, No. 4, 1977

A RESEARCH FRAMEWORK FOR KNOWLEDGE ACQUISITION TECHNIQUES AND TOOLS

John H. Boose
Knowledge Systems Laboratory, Advanced Technology Center
Boeing Computer Services, P.O. Box 24346
Seattle, Washington 98124 USA, john@boeing.com

ABSTRACT

Knowledge acquisition tools can be associated with knowledge-based application tasks and problem-solving methods. This descriptive approach provides a framework for analyzing and comparing tools and techniques, and focuses the task of building knowledge-based systems on the knowledge acquisition process. Knowledge acquisition research strategies discussed at recent Knowledge Acquisition Workshops are shown, distinguishing dimensions of knowledge acquisition tools are listed, and short descriptions of current techniques and tools are given.

Keywords: knowledge acquisition tools, knowledge acquisition, knowledge-based systems, knowledge engineering, problem-solving methods, task modeling.

APPLICATION TASKS AND PROBLEM-SOLVING METHODS

In this section, a framework for describing problem tasks and problem-solving methods is presented.

Several incompatible taxonomies exist for categorizing knowledge-based problem-solving application tasks. One scheme, illustrated below, divides them into *analysis* tasks and *synthesis* tasks. Analysis tasks involve identifying sets of objects based on their features. One characteristic of analysis tasks is that a complete set of solutions can be enumerated and included in the system. Synthesis (generative, or constructive) tasks require that a solution be built up from component pieces or subproblem solutions. In synthesis problem tasks there are too many potential solutions to enumerate and include explicitly in the system.

Analysis and synthesis tasks can be broken down into sub-task areas. One classification for this is shown here:

Analysis tasks
- Classification - categorizing based on observables.
- Debugging - prescribing remedies for malfunctions.
- Diagnosis - inferring system malfunctions from observables.
- Interpretation - inferring situation descriptions from sensor data.

Synthesis tasks
- Configuration - configuring collections of objects under constraints in relatively small search spaces.
- Design - configuring collections of objects under constraints in relatively large search spaces.
- Planning - designing actions.
- Scheduling - planning with strong time and/or space constraints.

Tasks combining analysis and synthesis
- Command and control - ordering and governing overall system control.
- Instruction - diagnosing, debugging, and repairing student behavior
- Monitoring - comparing observations to expected outcomes.
- Prediction - inferring likely consequences of given situations.
- Repair - executing plans to administer prescribed remedies.

Relationships exist between tasks and problem-solving methods For instance, the *heuristic classification* problem-solving method has been used for many knowledge based systems that solve analysis tasks (Clancey, 1986), and is employed in a variety of knowledge-based system development tools, or "shells" (S.1, M.1, EMYCIN, TI-PC, and so on). In heuristic classification, data is abstracted up through a problem hierarchy, problem abstractions are mapped onto solution abstractions, and solution abstractions are refined down through the solution hierarchy into specific solutions.

General methods for solving synthesis tasks (also *heuristic construction* tasks, Clancey, 1986) are sparse. Usually, a specific method is developed to solve a particular task, but it is difficult to generalize the method. Some form of directed backtracking is usually used to explore the problem space (see SALT and OPAL, below).

KNOWLEDGE ACQUISITION RESEARCH STRATEGIES

Knowledge acquisition techniques can be associated with specific tasks and specific problem-solving methods. Certain types of knowledge should be acquired to build the corresponding knowledge-based system. This idea was discussed at the First AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop held in Banff, Canada, in November, 1986 (Gaines and Boose, 1988). Builders of interactive knowledge acquisition tools were asked to try and classify their research and the research of others in terms of these relationships. Figure 1 shows a possible mapping of such relationships at a high level in the a *task classification hierarchy* and a *problem-solving method classification hierarchy*. Lower levels in the task hierarchy would be sub-tasks (i.e., trouble shooting and symptom analysis would be found under diagnosis), and the leaves of the task hierarchy would be specific application problems to be solved.

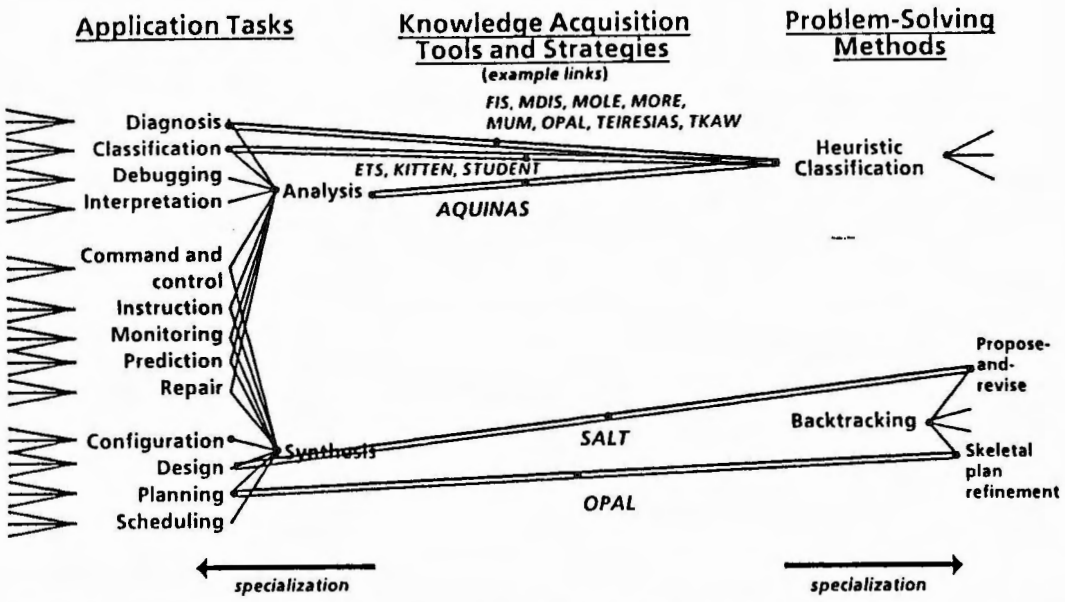


Figure 1. Knowledge acquisition tools may be associated with relationships between application tasks and problem-solving methods.

Knowledge acquisition tool research fell into several categories. Descriptions and references for the tools mentioned here are given later.

Research strategy 1:
Find and clarify knowledge acquisition strategies for a task-to-method relationship (usually a domain specific task employing a highly specialized method).
Examples for specific task domains ("bottom up") include OPAL, STUDENT, FIS, and MUM. Examples for general analysis tasks ("top down") include ETS, KITTEN, KRITON, AQUINAS, and KSS0.

Research strategy 2.
Pick a task, find and develop knowledge acquisition strategies for an applicable method, and then see if the method and strategies will generalize to another task.
Examples include ("middle out") TEIRESIAS, MDIS, MOLE, MORE, INFORM, KNACK, SALT, and TKAW.

Research strategy 3.
Develop languages for defining and describing tasks and methods.
Examples include the work of Bylander and Mittal (1986), Bylander and Chandrasekaran (1987), and Gruber and Cohen (1987).

Research strategy 4.
Build intelligent editors to help AI programmers construct large knowledge bases.
Examples include CYC, KREME, and NEXPERT.

Research strategy 4 was controversial. Some argued that knowledge should be tested for specific purposes as it is acquired rather than being constructed in a "use-vacuum." Proponents of the strategy stated that future expert systems will require these large knowledge bases to avoid problems of brittleness and narrowness of scope and to be able to perform analogy and discovery, and that tools to build and manage large knowledge bases should be developed now.

KNOWLEDGE ACQUISITION TOOLS: DIMENSIONS, TECHNIQUES, AND DESCRIPTIONS

Four Knowledge Acquisition for Knowledge-Based Systems workshops have been held since 1986 (see the bibliography for proceedings information):
- First AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, November, 1986.
- First European Knowledge Acquisition for Knowledge-Based Systems Workshop, Reading, England, September, 1987.
- Second AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, October, 1987.
- Second European Knowledge Acquisition for Knowledge-Based Systems Workshop, Bonn, Germany, June, 1988.

The next workshop will be:
- Third AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, November, 1988.

Representative knowledge acquisition techniques and tools presented at these workshops are described and analyzed below. First, an analysis of tools and tool dimensions is shown. Then, knowledge acquisition techniques and methods are listed and briefly described, along with computer-based tools, if any, that employ them. Finally, representative tools are listed, briefly described, and referenced.

Knowledge Acquisition Tool Dimensions

AQUINAS, a knowledge acquisition tool in use at The Boeing Company, was used prior to one of the Knowledge Acquisition Workshops to classify the tools and develop tool dimensions (references for AQUINAS appear below).

Knowledge in AQUINAS is represented, in part, in repertory grids. Objects appear along one axis of the grid and dimensions or traits appear along the other axis. AQUINAS helps the expert develop, analyze, refine, and test knowledge. A preliminary grid developed for the knowledge acquisition tools is shown in Figure 2.

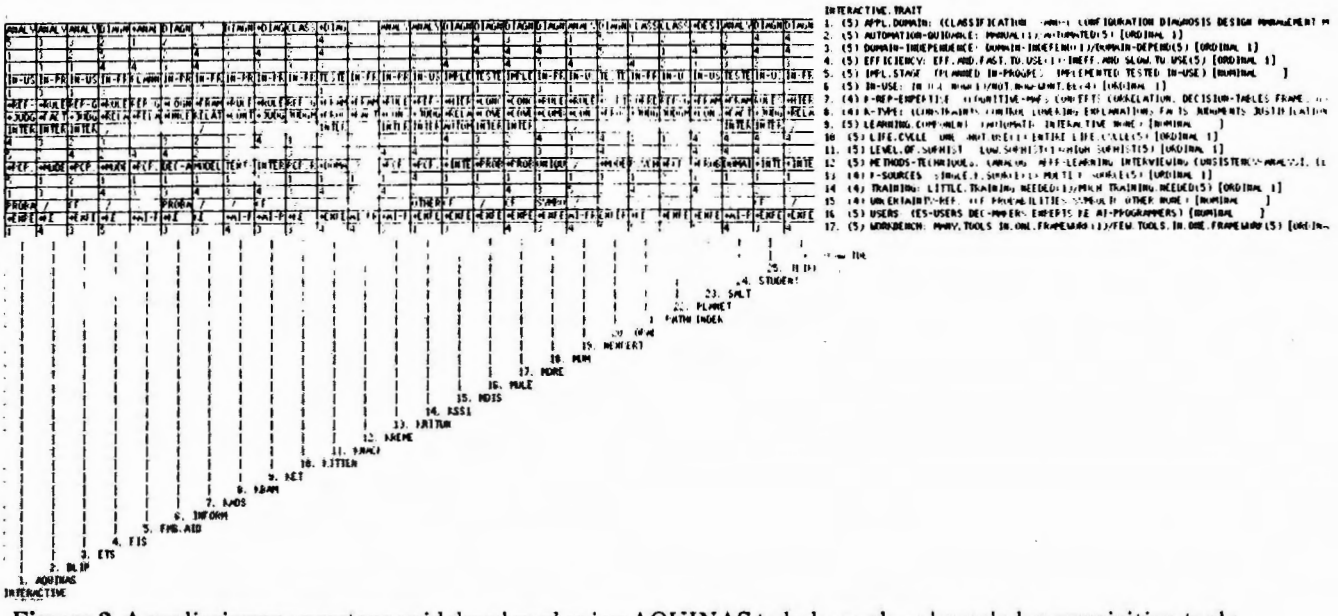


Figure 2. A preliminary repertory grid developed using AQUINAS to help analyze knowledge acquisition tools. Nominal as well as ordinal values are represented. *Distributions* of values exist when values in the grid are preceded by "*;" the highest value in the distribution is the one visible in the grid.

Eventually, AQUINAS elicited the following set of knowledge acquisition tool dimensions:

Application task dimensions

- Level of generality (domain dependence) - how domain-dependent is the tool?
- Analysis / synthesis - What broad categories of application tasks can the tool address?
- Specific task - Has the tool been built for a specific task? If so, what is the task?
- Application statistics (number and size of applications) - How many applications have been built with the tool? How diverse are they? How large are they? How much of the finished system did the tool help build?

Knowledge acquisition techniques and methods (these are listed separately below)

Utility dimensions

- Automated tool / semi-automated tool / manual technique - How much of the technique is implemented as a computer program? How "smart" is the tool? Is effective tool use dependent on the user, or does the tool offer semi-automated or automated assistance?
- Efficiency of use; speed of use - How hard is the tool to use? How efficient is knowledge elicitation and modeling? How well are the techniques implemented?
- Expertise representation method (cognitive maps, concepts, correlations, decision tables, frames, goal structures, hierarchies, operators, probability distributions, relations, repertory grids, rules, scripts, tables)
- High-level techniques / low-level techniques - How sophisticated are the techniques used?
- Implementation stage (planned, in progress, implemented,tested, in use, past use)
- Knowledge types (causal knowledge, classes, conceptual structures, constraints, control, covering, example cases, explanations, facts, goals, judgments, justifications, preferences, procedures, relations, spatial, strategic, temporal, terminology, uncertainties)
- Learning component (automatic, interactive, none) - If there is a learning component in the tool, how powerful is it? Is it automatic or interactive?
- Life cycle support (one-shot use to complete cycle support) - How much of the knowledge engineering and system delivery life cycle does the tool support?
- Multiple features / few features - How many techniques are integrated in a single framework? How well do multiple techniques support each other?
- Multiple knowledge sources support - Is there specific support for eliciting, analyzing, or delivering knowledge from multiple experts or other sources?
- Multiple knowledge views / few knowledge views - How many ways are there to look at elicited knowledge? What, if any, knowledge transformation techniques are employed?
- System in use / system not in use - Is the tool currently in use? Was the tool previously in use? Will the tool be in use in the future?
- Training needed - How much training is needed to use the tool? Can experts use the tool directly?

- Users (end-users of expert system, decision makers, experts, knowledge engineers, AI programmers needed) - Who are the targeted users of the tool?

Modeling dimensions

- Deep modeling / shallow modeling - Are "deep" models or "causal" models elicited?
- Multiple / single methods for handling uncertainty - What techniques, if any, are used to model uncertainty?

AQUINAS performed several analyses of the knowledge. An *implication analysis* produced by AQUINAS showed logical entailments between different dimensions (Figure 3) SINGLE.KNOWLEDGE.SOURCE implied FEW.TOOLS.IN.ONE.FRAMEWORK, for example. A *simliarity analysis* among *dimensions* showed, for example, that EFFICIENT.AND.FAST.TO.USE was closely coupled to LITTLE.TRAINING.NEEDED. A *simliarity analysis* among *tools* showed, for example, high similarity between ETS, KITTEN, and PLANET, and low similarity between FIS and KSS1 (similarity scores were produced for each pair of dimensions and each pair of tools). Other analyses were also performed.

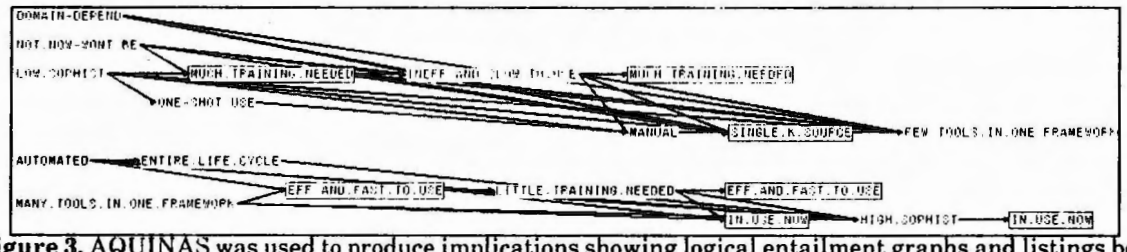


Figure 3. AQUINAS was used to produce implications showing logical entailment graphs and listings between tool dimensions.

AQUINAS also produced several "scatter tables" showing clusters of tools plotted on successive pairs of dimensions. Figure 4 (domain independence vs. task class) shows strong concentrations of knowledge acquisition tools for diagnostic tasks, but few knowledge acquisition tools for synthesis problems. Figure 5 shows that, generally, that domain-dependent tools exhibit a higher degree of process automation. Figure 6 shows that, generally, automated knowledge acquisition tools are easier to learn how to use. Figure 7 shows that knowledge acquisition tools that support more of the knowledge engineering life cycle tend to be more domain-dependent.

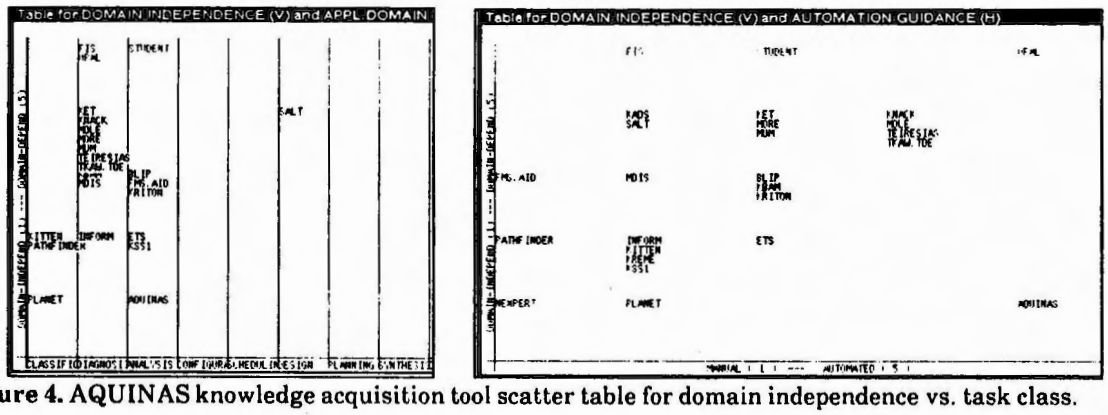


Figure 4. AQUINAS knowledge acquisition tool scatter table for domain independence vs. task class.

Figure 5. AQUINAS knowledge acquisition tool scatter table for domain dependence vs. degree of knowledge acquisition process automation.



Figure 6. AQUINAS knowledge acquisition tool scatter table for degree of KA process automation vs. amount of training needed to use tool effectively.

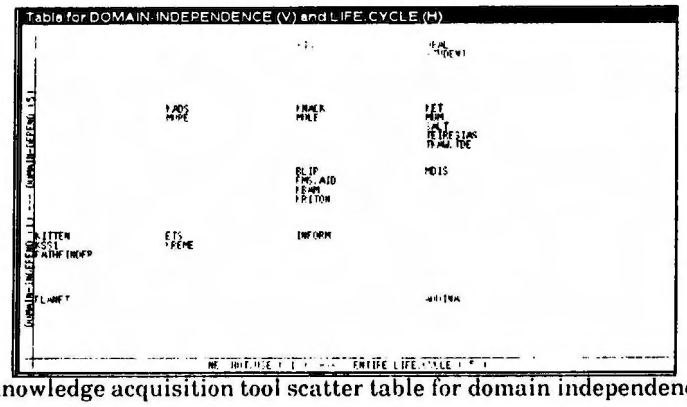


Figure 7. AQUINAS knowledge acquisition tool scatter table for domain independence vs. knowledge engineering life cycle support.

Other patterns in the tools are apparent. For instance, some tools try to draw power using strong specific domain knowledge (FIS, STUDENT, OPAL, MUM); other tools try to address a broader range of problems at the expense of some problem-solving power (ETS, KITTEN, AQUINAS). The few tools that address synthesis problems are domain dependent. Most researchers seem to be interested in applying their tools to more domain independent and/or harder tasks.

Knowledge Acquisition Techniques and Methods

In this section, knowledge acquisition methods and techniques presented at various Knowledge Acquisition Workshops are listed and briefly described. Techniques are divided into *manual* techniques and *computer-based* techniques. Computer-based techniques and tools are further divided into *interview-based, learning-based,* and miscellaneous techniques and tools. This section is meant to serve as a reference for those interested in obtaining more information about these techniques and tools.

1. Manual Techniques

Card Sorting - *sort objects on cards to help structure knowledge*
(Gammack and Young, 1984, in Welbank, 1987b)

Crawford Slip Method - *rapidly generate a large number of ideas*
(Rusk and Krone, 1984)

Focussed Interview - *interview with open questions and a list of topics to cover*
(Bradshaw, 1988), (Welbank, 1987b)

Overcoming Bias - *recognize and correct bias from knowledge sources*
(Cleaves, 1987) , (Moray, 1985), (Stephanou, 1987)

Participant Observation - *knowledge engineer becomes an apprentice or otherwise participates in the expert's problem-solving process*
(Welbank, 1987b)

Protocol Analysis (Case Walk-Through / Eidetic Reduction / Observation / Process-Tracing) - *record and analyze transcripts from experts thinking aloud during tasks*
(Belkin, Brooks, and Daniels, 1987) , (Breuker and Wielinga, 1987b), (Ericcson and Simon, 1984), (Gammack and Young, 1984, in Welbank, 1987b) , (Grover, 1983) , (Johnson, Nachtsheim, and Zualkernan, 1987) , (Killin and Hickman, 1986), (Littman, 1987), (Waldron, 1985)

Psychological Scaling (including multidimension scaling) - *use scaling techniques to help structure knowledge*
(Burton, Shadbolt, Hedgecock, and Rugg, 1987) , (Saaty, 1981), (Young and Gammack, 1987)

Structured Interview - *interview with strict agenda and list of specific questions relating to features of system*
(Bradshaw, 1988), (Freiling, Alexander, Messick, Rehfuss, and Shulman, 1985), (Slocombe, Moore, and Zelouf, 1986)

Teachback Interview - *knowledge engineer demonstrates understanding of expertise by paraphrasing or solving a problem*
(Johnson and Johnson, 1987)

Tutorial Interview - *expert delivers a lecture*
(Welbank, 1987b)

Uncertain Information Elicitation - *expert encodes uncertainty about the problem*
(Beyth-Marom and Dekel, 1985), (Hink and Woods, 1987), (Kahneman, Slovic, and Tversky, 1982), (Pearl, 1986), (Shafer and Tversky, 1985), (Spetzler and Stael von Holstein, 1983), (Stael von Holstein and Matheson, 1978), (Tversky, Sattah, and Slovic, 1987), (Wallsten and Budescu, 1983)

Unstructured Interview - *ask general questions and hope for the best, recording as much as possible*
(Kidd and Cooper, 1985), (Freeman, 1985), (Trimble, 1987), (Welbank, 1987a)

2. Computer-Based Techniques

When specific computer-based tools implement these methods, the name of the tool is listed. Computer-based tools are described and referenced in a following section. Work describing methods not implemented as computer-based tools are listed and referenced as a "Method."

a. Interview-Based and Miscellaneous Techniques

Automated / Mixed-initiative Interviewing - *the tool interviews the expert*
AQUINAS, ASK, CAP, ETS, KITTEN, KNACK, KRIMB, KRITON, KSS1, MDIS, MOLE, MORE, ODYSSEUS, PLANET, PROTOS, ROGET, SALT, TEIRESIAS, TKAW

Consistency Analysis - *analyze knowledge for consistency*
BLIP, FIS, KNAC, MUM, TEIRESIAS

Decision Analysis - *perform probabilistic inference and planning using influence diagrams*
INFORM
Methods - (Bradshaw and Boose, 1988), (von Winterfeldt and Edwards, 1986)

Delphi - *gather information from people independently*
MEDKAT

Domain Exploitation (Single Application) - *rely heavily on the domain for knowledge acquisition guidance*
FIS, LAS, LEAP, MUM, OPAL, STUDENT

Modeling *(deep models, causal models, cognitive models, task-level models) - use or generate models of the domain, possibly independent of a tool or a specific application*
ASTEK, BLIP, FIS, KADS, KRIMB, NEXPERT, ONTOS, OPAL
Methods - (Alexander, Freiling, Shulman, Rehfuss, and Messick, 1987), (Hayward, Wielinga, and Breuker, 1987), (Johnson, 1987), (Linster, 1987), (Morik, 1987b), (Regoczei and Plantinga, 1987), (Woods and Hollnagel, 1987), (Young and Gammack, 1987)

Multiple Source - *elicit and analyze knowledge from multiple sources separately and combine for use*
AQUINAS, ETS, MEDKAT, KITTEN, KSS1
Methods - (Gaines, 1987a, 1987b), (Mittal and Dym, 1985)

Performance System *(direct link or embedded) - generate knowledge that may be directly tested*
AQUINAS, ASK, BLIP, ETS, KAE, KNACK, KRITON, LEAP, MDIS, MOLE, MORE, MUM, ODYSSEUS, OPAL, ROGET, TEIRESIAS, TKAW, SALT

Physical Model Simulation - *use basic laws to derive physical models through simulation*
SIMULA

Problem-Solving Method Exploitation - *use information about the problem-solving method to guide knowledge acquisition*
AQUINAS, FMS-AID, KNACK, MDIS, MOLE, MORE, SALT, TEIRESIAS, TKAW

Protocol Analysis (Case Walk-Through / Eidetic Reduction / Observation / Process-Tracing) - *record and analyze transcripts from experts thinking aloud during tasks*
KRITON, MEDKAT

Psychological Scaling (including multidimension scaling) - *use scaling techniques to help structure knowledge*
AQUINAS, KITTEN, KSS1, PATHFINDER, PLANET
Methods - (Gaines and Shaw, 1981), (Kelly, 1955)

Repertory Grids / PCP - *use personal construct psychology and related methods to elicit and analyze knowledge*
AQUINAS, ETS, FMS-AID, KITTEN, KRITON, KSS1, PLANET

Textual Analysis / Natural Language Analysis - *generate knowledge directly by analyzing text*
KRITON, KSS1, KBAM

b. Learning-Based Techniques

Analogy - *apply knowledge from old situations in similar new situations*
CYC, TEIRESIAS

Apprenticeship Learning - *learn by watching experts solve problems*
ISG, LEAP, ODYSSEUS, PROTOS, OPAL, STUDENT

Example Selection - *select an appropriate set of examples for various learning techniques*
Methods - (Blythe, Corsi, and Needham, 1987), (Rissland, 1988)

Explanation-Based Learning - *deduce a general rule from a single example by relating it to an existing theory*
ACES, LAS, LEAP, OCCAM, ODYSSEUS
Method - (Kodratoff, 1987a, 1987b)

Decision Tree Induction / Analysis - *generate, analyze decision trees*
CART, ID3
Methods - (Bramer, 1987), (Goodman and Smyth, 1988), (Pettit and Pettit, 1987)

Genetic Algorithm - *genetic operators (crossing-over, mutation, inversion) are used to adapt a system's behavior*
Method - (Pettit and Pettit, 1987)

Induction of Models from Experience
AM, ATOM

Mimicking Expert Behavior
INDUCE

Performance Feedback - *performance feedback is used to reinforce behavior*
AQUINAS, MOLE, PERCEPTRON, PROTOS, STELLA

Rule / Knowledge Induction - *generate rules from other forms of knowledge*
AQ, AQUINAS, BLIP, ETS, KSS1, INSTIL
Methods - (Buntine, 1987), (Cleary, 1987), (Delgrande, 1987), (Goodman and Smyth, 1988), (Rissland, 1988), (Witten and MacDonald, 1988)

Similarity-Based Learning - *learn similarities from sets of positive examples and differences from sets of negative examples*
ID3, INSTIL

Systemic Principles Derivation - *use general principles to derive specific laws*
OBJ

Computer-Based Knowledge Acquisition Tools Descriptions and References

Knowledge acquisition tools presented and discussed above are briefly described and referenced here. Interactive tools in current use are described in terms of the task-method framework, intended user, uses, and features.

ACES - *learn heuristics for fault diagnosis from device descriptions using explanation-based learning* (Pazzani, 1987)
Task: Fault Diagnosis.
Method: Explanation-based learning.
Intended user: AI programmer.
Uses, systems built: Attitude control.
Features: heuristics examine atypical features and hypothesize potential faults; device models confirm or deny hypothesized faults.

AM - *induce models from experience* (Davis and Lenat, 1982)

AQ - *induce rules from sets of positive and negative training examples* (Michalski, 1983)

AQUINAS - *elicit and model information using a knowledge acquisition workbench including hierarchically-structured repertory grid-based interviewing and testing and other methods* (Boose and Bradshaw, 1987b), (Kitto and Boose, 1987), (Boose and Bradshaw, 1987a), (Boose, 1988), (Bradshaw and Boose, 1988), (Kitto and Boose, 1988), (Shema and Boose, 1988)
Task: Analysis.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Wide range of analytic tasks (decision aid for one-shot decisions, knowledge-based system development and delivery, group decision aid, shell front-end, situation insight and analysis).
Features: Workbench of tool kits: personal construct psychology methods; hierarchical structuring aids; induction; multiple expert analysis and reasoning; support for elicit, analysis, test, expand cycles.

ASK - *acquire strategic knowledge from experts using a justification language* (Gruber, 1988)
Task: Tasks where problem-solving strategy is important.
Method: Analysis.
Intended user: Expert, AI programmer.
Features: Addresses encoding examples, biasing generalization, new term problem.

ASTEK - *combine multiple paradigms for knowledge editing in a natural language discourse framework* (Jacobson and Freiling, 1988)

ATOM - *induce models from experience* (Gaines, 1977)

BLIP - *construct organized domain models automatically by learning from sloppy models* (Morik, 1987a), (Wrobel, 1988)
Task and method: Used to acquire problem-solving independent knowledge.
Intended user: Expert.
Features: "Sloppy modeling" of intermediate knowledge.

CART - *employ cross-validation to produce appropriately-sized decision trees* (Crawford, 1988)
Task: Classification.
Uses, systems built: Display fault classification, flower classification.
Features: Partitions measurement space into decision tree using concept induction.

CYC - *acquire and use knowledge through the use of analogy and a large existing knowledge base* (Lenat, Prakash and Shepard, 1986)
Task and method: Undefined.
Intended user: AI progammer.
Uses, systems built: Store "common sense" knowledge needed to understand encyclopedia articles.
Features: Classification of common sense knowledge primitives.

ETS - *interview experts using repertory grid-based methods and test the knowledge* (Boose, 1984, 1985, 1986a, b)
Task: Classification.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Wide range of small analytic system prototypes for knowledge engineering project start-up and feasibility analyses.
Features: Based on personal construct psychology (*repertory* or *rating grid* methods); embedded analysis, refinement, testing, and knowledge base shell generation.

FIS - *tie knowledge acquisition closely to the fault diagnosis domain* (De Jong, 1987)
Task: Mechanism diagnosis.
Method: Heuristic classification.
Intended user: AI programmer.
Uses, systems built: Fault isolation systems - radar units.
Features: Causal models built with replacable components, tests, attached rules; consistency check; rule induction.

FMS-AID - *combine repertory grid methods and Newell and Simon's problem space concept to manufacturing problems* (Garg-Janardan and Salvendy, 1987)

ID3 - *learn similarities and differences from training sets by optimizing global parameters* (Quinlan, 1983, 1987)

INDUCE - *mimic an expert's behavior* (Michalski and Chilausky, 1980)

INFORM - *elicit knowledge using decision analysis techniques* (Moore and Agogino, 1987)
Task: Mechanical diagnosis.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Model-based diagnosis of electro-mechanical systems.
Features: Decision analysis techniques - influence diagrams as knowledge representation.

INSTIL - *acquire knowledge using similarity-based learning combining aspects of both numeric and symbolic approaches* (Kodratoff and Manago, 1987a)

ISG - *link evidence to situations by synthesizing rules from interesting situations using an apprenticeship learning approach* (Wisniewski, Winston, Smith, and Kleyn, 1987)

Task: Classification.
Intended user: AI programmer, expert.
Uses, systems built: Geological formation analysis.
Features: Synthesize rules from situation-manifestation pairs by exploiting error propagation information.

KADS - *elicit knowledge decoupled from the design and implementation of the system* (Breuker and Wielinga, 1987a), (Anjewierden, 1987)

KAE - *capture scene analysis expertise* (Tranowski, 1988)

KBAM - *use natural language explanations to construct a domain-specific knowledge base* (Silvestro, 1988)
Task: Concept classification.
Method: Explanation-based learning.
Intended user: Expert.
Uses, systems built: Course descriptions.
Features: Natural language explanation analysis.

KET - *provide a graphical interface and analyze relationships to help experts write rules* (Esfahani and Teskey, 1987)
Task: Analysis.
Method: Heuristic classification.
Intended user: Expert, AI programmer.
Features: Elicitation of rules for the shell RBFS.

KITTEN - *interview experts using repertory grid-based methods* (Shaw and Gaines, 1987), (Shaw and Woodward, 1988)
Task: Classification.
Method: Heuristic classification (uses NEXPERT shell)
Intended user: Domain expert.
Uses, systems built: Range of small analytic system prototypes.
Features: Based on personal construct psychology (rating grid methods); text analysis to feed rating grids; linked to multi-user participant system.

KNAC - *use acquired assimilation knowledge to help enter new knowledge in a knowledge base* (Lefkowitz and Lesser, 1988)
Task: Analysis.
Method: Heuristic classification.
Intended user: Expert.
Features: Help integrate new knowledge into an existing POISE system knowledge base.

KNACK - *elicit and use knowledge about evaluation report generation* (Klinker, Bentolila, Genetet, Grimes, and McDermott, 1987), (Klinker, Genetet, and McDermott, 1988)
Task: Design evaluation and reporting.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Electro-mechanical system design evaluation and reporting for nuclear hardening.
Features: Gathers report outlines, phrases, and run-time procedures for filling in reports; generates shells that evaluate designs and produce reports.

KREME - *include multiple-representations in a knowledge editing environment* (Abrett and Burstein, 1987)
Task: Analysis.
Method: Heuristic classification.
Intended user: AI programmer.
Uses, systems built: Examine steam plant operations (STEAMER).
Features: Knowledge base editor; multiple views of knowledge.

KRIMB - *interview experts and build diagnostic domain models* (Cox and Blumenthal, 1987)
Task: Diagnosis.
Intended user: Expert.
Features: Intelligent questioning, reliability analysis of domain models.

KRITON - *combine repertory grid interviewing and protocol analysis to build knowledge at an intermediate level* (Diederich, Ruhmann, and May, 1987), (Diederich, Linster, Ruhmann, and Uthmann, 1987)
Task: Analysis.
Method: Heuristic classification.
Intended user: Domain expert, AI programmers.
Uses, systems built: Configuration of office equipment (in progress).
Features: Combines repertory grids, protocol analysis, content analysis; knowledge editor.

KSS0 - *elicit knowledge with a repertory grid-based interviewing tool including text analysis, behavior induction, and psychological scaling techniques* (Gaines, 1987a, 1987b, 1988), (Gaines and Sharp, 1987), (Shaw and Gaines, 1987)
Task: Analysis.
Method: Heuristic classification (uses NEXPERT shell).
Intended user: Expert.
Features: Workbench of tool kits: personal construct psychology methods; multiple expert analysis; text analysis, behavior induction, and psychological scaling.

LAS - *use apprenticeship learning to learn by watching experts solve problems* (Smith, Winston, Mitchell, and Buchanan, 1985)

LEAP - *use apprenticeship learning to learn steps in VLSI design by watching experts solve problems* (Mitchell, Mahadevan, and Steinberg, 1985)

MDIS - *experts are interviewed to describe mechanisms in a top-down structured manner for diagnostic problems* (Antonelli, 1983)
Task: Mechanical diagnosis.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Mechanical systems - jet engines.
Features: Top-down structured model elicitation, causal reasoning, simulation, multiple diagnostic strategies.

MEDKAT - *automate the Delphi technique to gather information from multiple experts* (Jagannathan and Elmaghraby, 1985).

MOLE - *exploit information about how problems are solved to elicit scarce diagnostic knowledge and use feedback to fine tune the knowledge* (Eshelman, Ehret, McDermott, and Tan, 1987), (Eshelman, 1988)
- Task: Mechanical diagnosis.
- Method: Heuristic classification.
- Intended user: Domain expert.
- Uses, systems built: Diagnosis - steel rolling mill, coal-burning power plant.
- Features: Distinguishing among and reasoning about covering, differentiating, and combining knowledge; understanding how to infer causal direction and support values.

MORE - *exploit information about how problems are solved to elicit extensive diagnostic knowledge* (Kahn, Nowlan, McDermott, 1985a,b)
- Task: Mechanical diagnosis.
- Method: Heuristic classification.
- Intended user: Domain expert.
- Uses, systems built: Diagnostic tasks.
- Features: Nets built around hypotheses, symptoms, and tests.

MUM - *evidential combination knowledge and control knowledge are elicited for medical problems* (Gruber and Cohen, 1987)
- Task: Medical diagnosis.
- Method: Heuristic classification.
- Intended user: AI programmer.
- Uses, systems built: Medical diagnosis; used with MU system.
- Features: Use of task-level primitives to capture domain knowledge at expert's level of abstraction

NEXPERT - *include multiple-representations in a knowledge editing environment with a performance component* (Rappaport, 1987), (Rappaport, 1988)
- Task: Analysis.
- Method: Heuristic classification.
- Intended user: AI programmer .
- Uses, systems built: Commercial shell.
- Features: Knowledge base editor, multiple knowledge views.

OBJ - *use general principles to derive specific laws* (Goguen and Meseguer, 1983)

OCCAM - *learn to predict outcomes of economic sanction episodes using explanation-based learning* (Pazzani, 1987)
- Task: Predicition.
- Method: Explanation-based learning.
- Intended user: AI programmer.
- Uses, systems built: Economic sanction episodes.
- Features: Generalize from the outcome of previous incidents.

ODYSSEUS - *refine and debug knowledge using apprenticeship learning techniques* (Wilkens, Clancey, and Buchanan, 1987)

ONTOS - *build domain models using cognitive and linguistic factors* (Monarch and Nirenburg, 1987)

OPAL - *tie knowledge acquisition closely to the cancer treatment domain* (Musen, Fagan, Combs, and Shortliffe, 1987)
- Task: Medical diagnosis.
- Method: Skeletal plan refinement.
- Intended user: Domain expert.
- Uses, systems built: Cancer therapy management (in ONCOCIN).
- Features: Use semantics of application domain (vs. knowledge structure or problem-solving method) to manage knowledge acquisition.

PATHFINDER - *use psychological scaling techniques to help structure knowledge hierarchically* (Cooke and McDonald, 1987)

PLANET - *use repertory grids for psychological interviewing and analysis* (Shaw, 1984), (Gaines, and Shaw, 1986)
- Task: Classification.
- Uses, systems built: Used by psychologists and others for computer-based repertory grid elicitation.
- Features: Repertory grid elicitation and analysis.

PROTOS - *use exemplar-based learning in an apprenticeship learning system* (Bareiss, Porter, and Wier, 1988)
- Task: Analysis.
- Method: Heuristic classification.
- Intended user: Expert.
- Uses, systems built: Clinical audiology.
- Features: Use inductive and deductive techniques to retain, index, and match exemplars.

ROGET - *interview experts and produce conceptual structures of the domain* (Bennet, 1985)
- Task: Medical Diagnosis.
- Method: Heuristic Classification.
- Intended user: Expert.
- Uses, systems built: Used to generate knowledge bases for EMYCIN.
- Features: Elicits conceptual model from the expert by asking for concepts and support relationships.

SALT - *elicit and deliver knowledge for constructive constraint satisfaction tasks* (Marcus, McDermott, and Wang, 1985), (Marcus, 1987), (Stout, Caplain, Marcus, and McDermott, 1988)
- Task: Configuration, scheduling.
- Method: Propose-and-revise (directed backtracking).
- Intended user: Domain expert.
- Uses, systems built: Physical system configuration (elevator configuration); simulation lab resource scheduling.
- Features: Expert knowledge consists of parts or modules, constraints, and what to do when constraints are violated.

SIMULA - *use basic laws to derive physical models through simulation* (Nygaard and Dahl, 1981)

STELLA - *performance feedback is used to reinforce behavior* (Gaines and Andreae, 1966)

STUDENT - *tie knowledge acquisition closely to the statistical consulting domain* (Gale, 1987)
Task: Statistical analysis classification.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: Statistical consulting.
Features: Domain primitives (frames) that specify what information is relevant to data analysis.

TEIRESIAS - *model existing knowledge to monitor refinements and help debug consultations* (Davis and Lenat, 1982)
Task: Medical diagnosis.
Method: Heuristic classification.
Intended user: Domain expert.
Uses, systems built: MYCIN, systems built with EMYCIN.
Features: Rule-based repair through errors of commission and omission; suggestion of new rules based on statistical models.

TKAW/TDE - *exploit information about how problems are solved to elicit trouble-shooting knowledge* (Kahn, Breaux, Joeseph, and DeKlerk, 1987)
Task: Mechanical diagnosis.
Method: Heuristic classification.
Intended user: Domain expert, AI programmer.
Uses, systems built: Trouble-shooting electro-mechanical systems.
Features: Mixed-initiative environment, concentrates on search control in trouble-shooting tasks.

DISCUSSION

The first part of this paper presented a framework for associating knowledge acquisition tools with knowledge-based application tasks and problem-solving methods. Specific tools and tool classes were associated with specific application tasks and methods (i.e., AQUINAS is associated with analysis tasks and the heuristic problem-solving method; SALT is associated with configuration and design applications and a specialized form of backtracking).

These associations help define the depth and breadth of current knowledge acquisition research. Our research using AQUINAS has led us to try and build a broad link (multiple integrated tool sets) between a general application task class (analysis problems) and a powerful problem-solving method (heuristic classification). Other successful work has led researchers to tightly couple knowledge acquisition tools to a domain task (for example, FIS, STUDENT, OPAL).

Current research strategies were mentioned. Associations in the task-method framework where no tools exist can point out promising areas for new research. Can special forms of knowledge acquisition tools be associated with debugging tasks and heuristic classification, or planning and new specializations of backtracking?

This descriptive approach provides a framework for studying and comparing tools. It also emphasizes the need for a more refined application task hierarchy, and the need to recognize and generalize new problem-solving methods.

The rest of the paper analyzed dimensions and relationships among the tools and techniques, and classified and briefly described current knowledge acquisition tools and methods. It is hoped that this reference work will be useful both as an introduction and to those are pursue this field of research.

ACKNOWLEDGEMENTS

Thanks to Tom Addis, Miroslav Benda, Jeff Bradshaw, Stan Covington, Brian Gaines, Linda Green, Cathy Kitto, Marc Linster, Sandy Marcus, Art Nagai, Peter Russo, Doug Schuler, Mildred Shaw, David Shema, Lisle Tinglof-Boose, and the participants from the Knowledge Acquisition Workshops for their contributions and support. AQUINAS was developed at the Knowledge Systems Laboratory, Boeing Advanced Technology Center, Boeing Computer Services, in Seattle, Washington.

BIBLIOGRAPHY

Proceedings from Knowledge Acquisition Workshops

The *Proceedings of the First AAAI-sponsored Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Canada, November 2-6, 1986, were published in the *International Journal of Man-Machine Studies* in the January, February, April, August, and September issues in 1987. A two-volume Academic Press book containing the proceedings should be available in 1988.

The *Proceedings of the Second AAAI-sponsored Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Canada, October 19-23, 1987, will be published in the *International Journal of Man-Machine Studies* early in 1988, and in an Academic Press book later in 1988.

The *Proceedings of the First European Workshop on Knowledge Acquisition*, Reading, U.K, September 1-3, 1987, will be published in an Academic Press book in 1988. Some copies of the proceedings distributed at the workshop may still be available directly from Tom Addis, Department of Computer Science, Reading University, Whiteknights, PO Box 220, Reading RG6 2AX, U.K. (Tom Addis@reading.ac.uk).

The *Proceedings of the Second European Workshop on Knowledge Acquisition*, Bonn, June 19-32, 1988, will probably be published in an Academic Press book in late 1988. Some copies of the proceedings distributed at the workshop may be available directly from Marc Linster, Institut for Applied Information Technology, German Research Institut for Mathematics and Dataprocessing, Schloss Birlinghoven, Postfach 1240, St. Augustin 1, Germany D-5205.

The *Proceedings of the Third AAAI-sponsored Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Canada, November 7-11, 1988, will probably be published in the *International Journal of Man-Machine Studies* early in 1989, and in an Academic Press book later in 1989.

References

Abrett, G., Burstein, M. H., (1987).The KREME Knowledge Editing Environment, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.

Alexander, J. H., Freiling, M. J., Shulman, S. J., Rehfuss, S., Messick, S. L. (1987). Ontological Analysis: An Ongoing Experiment, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.

Anjewierden, A. (1987). The KADS System, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Antonelli, D. (1983). The Application of Artificial Intelligence to a Maintenance and Diagnostic Information System (MDIS), in the *Proceedings of the Joint Services Workshop on Artificial Intelligence in Maintenance*, Boulder, CO.

Bareiss, E. R., Porter, B. W., Wier, C. C. (1988). Protos: An Exemplar-Based Learning Apprentice, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Belkin, N. J., Brooks, H. M., Daniels, P. J. (1987). Knowledge Elicitation Using Discourse Analysis, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.

Bennet, J. S. (1985). A Knowledge-Based System for Acquiring the Conceptual Structure of a Diasgnostic Expert System, Journal of Automated Reasoning, 1, 49-74.

Beyth-Marom, R. and Dekel, S. (1985). An Elementary Approach to Thinking Under Uncertainty, translated and adapted by S. Lichtenstein, B. Marom, and R. Beyth-Marom, Hillsdale, N.J.: Lawrence Erlbaum.

Blythe, J., Corsi, P., and Needham, D. (1987). An Experimental Protocol for the Acquisition of Examples for Learning, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Boose, J. H. (1984). Personal Construct Theory and the Transfer of Human Expertise, in the *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)*, Austin, Texas.

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

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

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

Boose, J. H. (1988). Uses of Repertory Grid-Centered Knowledge Acquisition Tools for Knowledge-Based Systems, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Boose, J. H., and Bradshaw, J. M. (1987a). Expertise Transfer and Complex Problems: Using Aquinas as a Knowledge Acquisition Workbench for Expert Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.

Boose, J. H., and Bradshaw, J. M. (1987b). AQUINAS: A Knowledge Acquisition Workbench for Building Knowledge-Based Systems, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Bradshaw, J. M. (1988). Strategies for Selecting and Interviewing Experts, Boeing Computer Services Technical Report, in preparation.

Bradshaw, J. M., and Boose, J. H. (1988). Decision Analytic Techniques for Knowledge Acquisition: Combining Situation and Preference Models Using Aquinas, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Bramer, M. (1987). Automatic Induction of Rules from Examples: A Critical Analysis of the ID3 Family of Rule Induction Systems, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Breuker, J., Wielinga, B. (1987). Knowledge Acquisition as Modeling Expertise: The KADS Methodology, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Breuker, J., Wielinga, B. (1987b). Use of Models in the Interpretation of Verbal Data, in A. Kidd, Knowledge Elicitation for Expert Systems: A Practical Handbook, Plenum Press.

Buntine, W. (1987). Induction of Horn Clauses: Methods and the Plausible Generalisation Algorithm, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.

Burton, A. M., Shadbolt, N. R., Hedgecock, A. P., Rugg, G. (1987). A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Bylander, T., Chandrasekaran, B. (1987). Generic Tasks in Knowledge-Based Reasoning: The 'Right' Level of Abstraction for Knowledge Acquisition, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.

Bylander, T., and Mittal, S. (1986). CSRL: A Language for Classificatory Problem-Solving and Uncertainty Handling, *AI Magazine*, August.

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

Cleary, J. G. (1987). Acquisition of Uncertain Rules in a Probabilistic Logic, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.

Cleaves, D. A. (1987). Cognitive Biases and Corrective Techniques; Proposals for Improving Elicitation Procedures for Knowledge-Based Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.

Cooke, N. M., McDonald, J. E. (1987). The Application of Psychological Scaling Techniques to Knowledge Elicitation for Knowledge-Based Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.

Cox, L. A., Blumenthal, R. (1987). KRIMB: An Intelligent Knowledge Acquisition and Representation Program for Interactive Model Building, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Crawford, S. L. (1988). Extensions to the CART Algorithm, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine* Studies, in press.

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

De Jong, K. (1987). Knowledge Acquisition for Fault Isolation Expert Systems, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.

Delgrande, J. P. (1987). A Formal Approach to Learning From Examples, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.

Diederich, J., Ruhmann, I., and May, M. (1987). KRITON: A Knowledge Acquisition Tool for Expert Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.

Diederich, J., Linster, L., Ruhmann, I., and Uthmann, T. (1987). A Methodology for Integrating Knowledge Acquisition Techniques, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, University of Reading, September.

Ericcson, K. A., Simon, H. A. (1984). Protocol Analysis: Verbal Reports as Data, MA: The MIT Press.

Esfahani, L., Teskey, F. N. (1987). KET, A Knowledge Encoding Tool, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

Eshelman, L, Ehret, D., McDermott, J., and Tan, M. (1987). MOLE: A Tenacious Knowledge Acquisition Tool, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.

Eshelman, L (1988). MOLE: A Knowledge Acquisition Tool That Buries Certainty Factors, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Freeman, P. (1985). Knowledge Elicitation - A Commercial Perspective, *Proceedings of the 1st International Expert System Conference*, London, October.

Freiling, M., Alexander, J., Messick, S., Rehfuss, S., and Shulman, S. (1985). Starting a knowledge engineering project: A step-by-step approach. *AI Magazine*, Fall, 150-164.

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, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.
Gaines, B. R. (1987b). Knowledge Acquisition for Expert Systems, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Gaines, B. R. (1987c). How Do Experts Acquire Expertise? *Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge-based Systems*, Reading, England, September 1987.
Gaines, B. R. (1988). Advanced Expert System Support Environments, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Gaines, B. R., and Andreae, J. H. (1966). A Learning Machine in the Context of the General Control Problem, *Proceedings of the 3rd Congress of the International Federation for Automatic Control*, London: Butterworths.
Gaines, B. R., and Boose, J. H. (1988). A Summary of the AAAI-sponsored Knowledge Acquisition for Knowledge-Based System Workshops, *AI Magazine*, in press.
Gaines, B. R., Sharp, M. (1987). A Knowledge Acquisition Extension to Notecards, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Gaines, B. R., and Shaw, M. L. G. (1981). 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.
Gaines, B. R., and Shaw, M. L. G. (1986). Interactive Elicitation of Knowledge from Experts, *Future Computing Systems*, 1(2).
Gale, W. A. (1987). Knowledge-Based Knowledge Acquisition for a Statistical Consulting System, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.
Gammack, J. G., and Young, R. M. (1984). Psychological Techniques for Eliciting Expert Knowledge, in R&D in Expert Systems, Proceedings of the 4th Expert System Conference, Warwick ed Max Bramer, CUP.
Garg-Janardan, C., and Salvendy, G. (1987). A Conceptual Framework for Knowledge Elicitation, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.
Goguen, J. A., and Meseguer, J. (1983). Programming with Parametrized Abstract Objects in OBJ, Theory and Practice of Programming Technology, Amsterdam: North-Holland.
Goodman, R. M. F., Smyth, P. (1988). Learning from Examples Using Information Theory, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Grover, M. D. (1983). A Pragmatic Knowledge Acquisition Methodology, *Proceedings of the Eighth International Conference of Artificial Intelligence*, 436-438, Karlsrue, West Germany.
Gruber, T. R. (1988). Acquiring Strategic Knowledge from Experts, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Gruber, T. R., Cohen, P. R. (1987). Design for Acquisition: Principles of Knowledge System Design to Facilitate Knowledge Acquisition, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.
Hayward, S. A., Wielinga, B. J., Breuker, J. A. (1987). Structured Analysis of Knowledge, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.
Hink, R. F. & Woods, D. L. (1987). How humans process uncertain knowledge: An introduction for knowledge engineers. *AI Magazine*, Fall, 41-53.
Ivey, A. E., & Simek-Downing, L. Counseling and Psychotherapy: Skills, Theories, and Practice. Englewood Cliffs, N.J.: Prentice-Hall, 1980.
Jacobson, C., Freiling, M. J. (1988). ASTEK: A Multi-Paradigm Knowledge Acquisition Tool for Complex Structured Knowledge, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Jagannathan, V., and Elmaghraby, A. S. (1985). MEDKAT: Multiple Expert Delphi-Based Knowledge Acquisiton Tool, Engineering Mathematics and Computer Science Department Technical Report, University of Louisville.
Johnson, L. and Johnson, N. E. (1987). Knowledge Elicitation Involving Teachback Interviewing, in A. Kidd (ed.), Knowledge Elicitation for Expert Systems: A Practical Handbook, New York: Plenum Press.
Johnson, N. (1987). Mediating Representations in Knowledge Elicitation, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Johnson, P. E., Zaulkernan, I., Garber, S. (1987). Specification of Expertise: Knowledge Acquisition for Expert Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.
Kahn, G., Nowlan, S., McDermott, J. (1985a). Strategies for Knowledge Acquisition, IEEE Transactions of Pattern Analysis and Machine Intelligence, PAMI-7(3), 5111-522.
Kahn, G., Nowlan, S., and McDermott, J. (1985b). MORE: An Intelligent Knowledge Acquisition Tool, in the *Proceedings of the Ninth Joint Conference on Artificial Intelligence*, Los Angeles, CA, August.
Kahn, G. S., Breaux, E. H., Joeseph, R. L., and DeKlerk, P. (1987). An Intelligent Mixed-Initiative Workbench for Knowledge Acquisition, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, in press.
Kahneman, D., Slovic, P., & Tversky, A. (Eds.)(1982). Judgment Under Uncertainty: Heuristics and Biases. Cambridge: Cambridge University Press.
Kelly, G. A. (1955). The Psychology of Personal Constructs, New York: Norton.
Kidd, A. L. (1987). Knowledge Acquisition for Expert Systems: A Practical Handbook. New York: Plenum Press.
Kidd, A. L. and Cooper, M. B. (1985). Man-Machine Interface Issues in the Construction and Use of an Expert System, *International Journal of Man-Machine Studies*, 22.
Killin, J. L., and Hickman, F. R. (1986). The Role of Phenomenological Techniques of Knoweldge Elicitation in Complex Domains, Expert Systems '86, Brighton.
Kitto, C. M., and Boose, J. H. (1987). Heuristics for Expertise Transfer: The Automatic Management of Complex Knowledge Acquisition Dialogs, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No. 2.
Kitto, C. M., and Boose, J. H. (1988). Selecting Knowledge Acquisition Tools and Strategies Based on Application Characteristics, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Klinker, G., Bentolila, J., Genetet, S., Grimes, M., and McDermott, J. (1987). KNACK: Report-Driven Knowledge Acquisition, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.
Klinker, G., Genetet, S., McDermott, J. (1988). Knowledge Acquisition for Evaluation Systems, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Kodratoff, Y., Manago, M. (1987a). Generalization in a Noisy Environment: The Need to Integrate Symbolic and Numeric Techniques in Learning, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.
Kodratoff, Y., (1987b). Machine Learning and Explanations, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Lefkowitz, L. S., Lesser, V. R. (1988). Knowledge Acquisition as Knowledge Assimilation, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Lenat, D., Prakish, M., Shepard, M. (1986). CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks, *AI MAgazine* 6(4), 65-85.
Linster, M. (1987). On Structuring Knowledge for Incremental Knowledge Acquisition, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Littman, D. C. (1987). Modeling Human Expertise in Knowledge Engineering: Some Preliminary Observations, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.
Marcus, S., McDermott, J., and Wang, T. (1985). Knowledge Acquisition for Constructive Systems, in the *Proceedings of the Ninth Joint Conference on Artificial Intelligence*, Los Angeles, CA, August.
Marcus, S. (1987). Taking Backtracking with a Grain of SALT, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No. 4.
Michalski, R. S., Chilausky, R. L. (1980). Knowledge Acquisiton by Encoding Expert Rules versus Computer Induction from Examples - A Case Study Involving Soybean Pathology. *International Journal of Man-Machine Studies*, 12, 63-87.
Michalski, R. S. (1983). Theory and Methodology of INductive Learning, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Machine Learning, an Artificial Intelligence Approach, Tioga Publishing Company.
Mitchell, T. M., Mahadevan, S., and Steinberg, Louis I. (1985). LEAP: A Learning Apprentice for VLSI Design, *Proceedings of the International Conference on Atrificial Intelligence*, 1985, Los Angeles.
Mittal, S., & Dym, C. Knowledge acquisition from multiple experts. *AI Magazine*, Summer 1985, 32-36

Monarch, I., Nirenburg, S. (1987). The Role of Ontology in Concept Acquisition for Knowledge-Based Systems, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Moore, E. A., Agogino, A. M. (1987). INFORM: An Architecture for Expert-Directed Knowledge Acquisition, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.
Moray, N., Sources of Bias and Fallibility in Humans, Workshop on Knowledge Engineering in Industry, University of Toronto, May, 1985.
Morik, K. (1987a). Acquiring Domain Models, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.
Morik, K. (1987b). Knowledge Acquisition and Machine Learning - The Issue of Modeling, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Musen, M. A., Fagan, L. M., Combs, D. M., and Shortliffe, E. H. (1987). Use of a Domain Model to Drive an Interactive Knowledge-Editing Tool, special issue on the 1st AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.
Nygaard, K., Dahl, O. J. (1981). The Development of the SIMULA Languages, in R. L. Wexelblat (ed.), History of Programming Languages, 439-480, New York: Academic Press.
Pazzani, M. J. (1987). Explanation-Based Learning for Knowledge-based Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.
Pearl, J., Fusion, Propagation and Structuring in Belief Networks, Technical Report CSD-850022, R-42-VI-12, Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA, April, 1986.
Pettit, E. J., Pettit, M. J. (1987). Analysis of the Performance of a Genetic Algorithm-Based System for Message Classification in Noisy Environments, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, Vol. 27, No. 2.
Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, (J. H. Boose and B. R. Gaines, eds.) Banff, Canada, November, 1986.
Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, (J. H. Boose and B. R. Gaines, eds.) Banff, Canada, October, 1987.
Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems, (T. Addis, J. H. Boose and .B R. Gaines, eds.) Reading, England, September, 1987.
Proceedings of the Second European Workshop on Knowledge Acquisition for Knowledge-Based Systems, (M. Linster, J. H. Boose and B. R. Gaines, eds.) Bonn, Germany, June, 1988.
Quinlan, J. R. (1983). Learning Efficient Classification Procedures and Their Application to Chess End Games, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Machine Learning, an Artificial Intelligence Approach, Tioga Publishing Company.
Quinlan, J. R. (1987). Simplifying Decision Trees, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 5, *International Journal of Man-Machine Studies*, Vol 27, No. 3.
Rappaport, A. (1987). Multiple Problem Spaces in the Knowledge Design Process, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No.4.
Rappaport, A. (1988). Cognitive Primitives, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Regoczei, S., Plantinga, E. P. O. (1987). Ontology and Inventory: A Foundation for a Knowledge Acquisition Methodology, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, in press.
Rissland, E. L. (1988). The Problem of Intelligent Example Selection, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Rusk, R. A., Krone, R. M. (1984). The Crawford Slip Method (CSM) as a Tool for extraction of Expert Knowledge, in G. Salvendy (ed.), Human-Computer Interaction, 279-282, New York: Elsevier.
Saaty, T. L. (1981). The Analytic Hierarchy Process, New York: McGraw-Hill.
Shafer, G., & Tversky, A. Languages and designs for probability judgment. *Cognitive Science*, 9, 1985, 309-339.
Shalin, V. L., Wisniewski, E. J., Levi, K. R., Scott, P. D. (1988). A Formal Analysis of Machine Learning Systems for Knowledge Acquisition, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Shaw, M. L. G. (1984). Interaction Knowledge Elicitation, *Proceedings of the Canadian Information Processing Society Annual Conference*, Calgary, Canada.
Shaw, M. L. G., and Gaines, B. R. (1987). Techniques for Knowledge Acquisition and Transfer, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 5, *International Journal of Man-Machine Studies*, Vol 27., No. 3.
Shaw, M. L. G., Woodward, J. B. (1988). Validation of a Knowledge Support System, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Shema, D. B., and Boose, J. H. (1988). Refining Problem-Solving Knowledge in Repertory Grids Using a Consultation Mechanism, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Silvestro, K. (1988). An Explanation-Based Approach to Knowledge-Base Acquisition, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Slocombe, S., Moore, K. D. M., and Zelouf, M. (1986). Engineering Expert System Applications, Expert Systems-86, December, Brighton.
Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. G. (1985). Representaion and Use of Explicit Justifications for Knowledge Base Refinements, *Proceedings of the Internation Joint Conference on Artificial Intelligence*, Los Angeles, CA.
Spetzler, C., and Stael von Holstein, C. (1983). Probability Encoding in Decision Analysis, in R. Howard and J. Matheson (eds.), Readings on the Principles and Applications of Decision Analysis, Volume 2, Palo Alto, CA: Strategic Decisions Group.
Stael von Holstein, C. & Matheson, J.E. A manual for encoding probability distributions. Menlo Park, CA: SRI International, 1978.
Stephanou, H. Perspectives on imperfect information processing. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987, 17, 780-798.
Stout, J., Caplain, G., Marcus, S., McDermott, J (1988). Toward Automating Recognition of Differing Problem-Solving Demands, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Tranowski, D. (1988). A Knowledge Acquisition Environment for Scene Analysis, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Trimble, G., Cooper, C. N. (1987). Experience of Knowledge Acquisition for Expert Systems in Construction, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Tversky, A., Sattah, S., & Slovic, P. (1987). Contingent weighting in judgment and choice. Menlo Park, California: Stanford University.
Waldron, V.R. (1985). Process Tracing as a Method for Initial Knowledge Acquisition, *Proceedings of the 2nd IEEE Conference on AI Applications*, Florida, December.
Wallsten, T., and Budescu, D., Encoding Subject Probabilities: A Psychological and Psychometric Review, *Management Science*, 29:2, 1983.
Welbank, M. (1987a). Knowledge Acquisition: A Survey and British Telecom Experience, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.
Welbank, M. (1987b). Knowledge Acquisition Update, Insight Study No. 5, Systems Designers, England.
Wilkens, D. C., Clancey, W. J., and Buchanan, G. G. (1987). KNnowledge Base Refinement by Editing Abstract Control Knowledge, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 5, *International Journal of Man-Machine Studies*, Vol. 27, No. 4.
von Winterfeldt, D., and Edwards, W. (1986). Decision Analysis and Behavioral Research. Cambridge: Cambridge University Press.
Wisniewski, E., Windston, H., Smith, R., Kleyn, M. (1987). Case Generation for Rule Synthesis, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 4, *International Journal of Man-Machine Studies*, in press.
Witten, I. H., MacDonald, B. (1988). Using Concept Learning for Knowledge Acquisition, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Woods, D. D., Hollnagel, E. (1987). Mapping Cognitive Demands and Activities in Complex Problems Solving Worlds, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No.2.
Wrobel, S. (1988). Design Goals for Sloppy Modeling Systems, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.
Young, R. M., Gammack, J. (1987). Role of Psychological Techniques and Intermediate Representations in Knowledge Elicitation, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University, September.

KNOWLEDGE ACQUISITION BY OBSERVATION : APPLICATION TO INTELLIGENT TUTORING SYSTEMS

Guy A. BOY and Nicole NUSS
Groupe d'Intelligence Artificielle
Centre d'Etudes et de Recherches de Toulouse
Office National d'Etudes et de Recherches Aérospatiales
2, avenue Edouard Belin
31055 Toulouse Cedex
France

Abstract. This paper presents an architecture for an Intelligent Tutoring System (ITS) based on a pedagogic model. This model takes into account the student's behavior in a "short term supervisory control loop" and improves the initial pedagogic knowledge base in a "long term evaluation loop". Observation is the key factor in this approach. It allows incremental knowledge acquisition from an initial pedagogic knowledge base. A learning technique by Specialisation-Structuration has been developed and is presented here. This approach is being applied in instruction of aircraft pilots.

Keywords: Intelligent Tutoring Systems, Knowledge Acquisition, Observer, Pedagogic Model, Complex Environment, Global Approach.

1. Introduction

Intelligent Tutoring Systems (ITSs) are programmed teaching-aids grounded in artificial intelligence (AI) (Sleeman & Brown, 1982). AI techniques can be used for improving computer-assisted instruction in three areas : (1) improving student-computer interaction, (2) mastering domain knowledge through problem solving, and (3) taking into account the student's model and understanding of his/her behavior. Our work is more focused on area 3. Conventional tutoring systems use very little pedagogic knowledge and need to improve the modularity of their knowledge bases (KBs) (Forcheri & Molfino, 1986). The major critical problem in KB systems development is the knowledge acquisition process (Freidland, 1981; Feigenbaum, 1982; Freiling et al., 1985; Gevarter, 1984; Quinlan, 1984). This is particularly so in dynamic and moving environments. In the present knowledge acquition technique, the initial KB includes technical and pedagogic knowledge. Technical knowledge is dependent on the domain to be learnt. Pedagogic knowledge has to be student-dependant according to on-line observation. It can be modelled as a meta level of the technical knoweldge. As pedagogic knowledge acquisition is a major limiting step, it will be the focus of attention for the following technique.

With this more global approach, it is possible to tackle complex domains. For instance, our domain is flight management instruction. Our students are transport pilots. In this complex situation, the main goal is design a pedagogic model. This new approach in knowledge acquisition comes from a general concept of operator assistant (OA) systems (Boy, 1988a). OA systems are both computer aids and observers of the environment, including the system being controlled and the human operator himself. In the present work, we will talk about "student assistant system". This paper presents a pedagogic model which takes into account two types of information acquisition from the environment. On the one hand, the short term supervisory control loop allows the system to react to observed student behavior, i.e., to correct immediate errors. On the other hand, the long term evaluation loop allows modification of the knowledge base from observation and interpretation of the obtained results.

As the topic of this paper is concerned with intelligent tutoring systems and "expert system" knowledge acquisition, we will differenciate between human learning and machine learning to avoid any confusion.

2. Pedagogic Model

Good professors are also good observers of the behavior of their students. They are able to analyze and propose adapted pedagogic situations to offer their students an easier problem to solve. This constitutes the main issue in the present approach. Our pedagogic model (Figure 1) includes two loops: (1) a short term supervisory control loop, and (2) a long term evaluation loop.

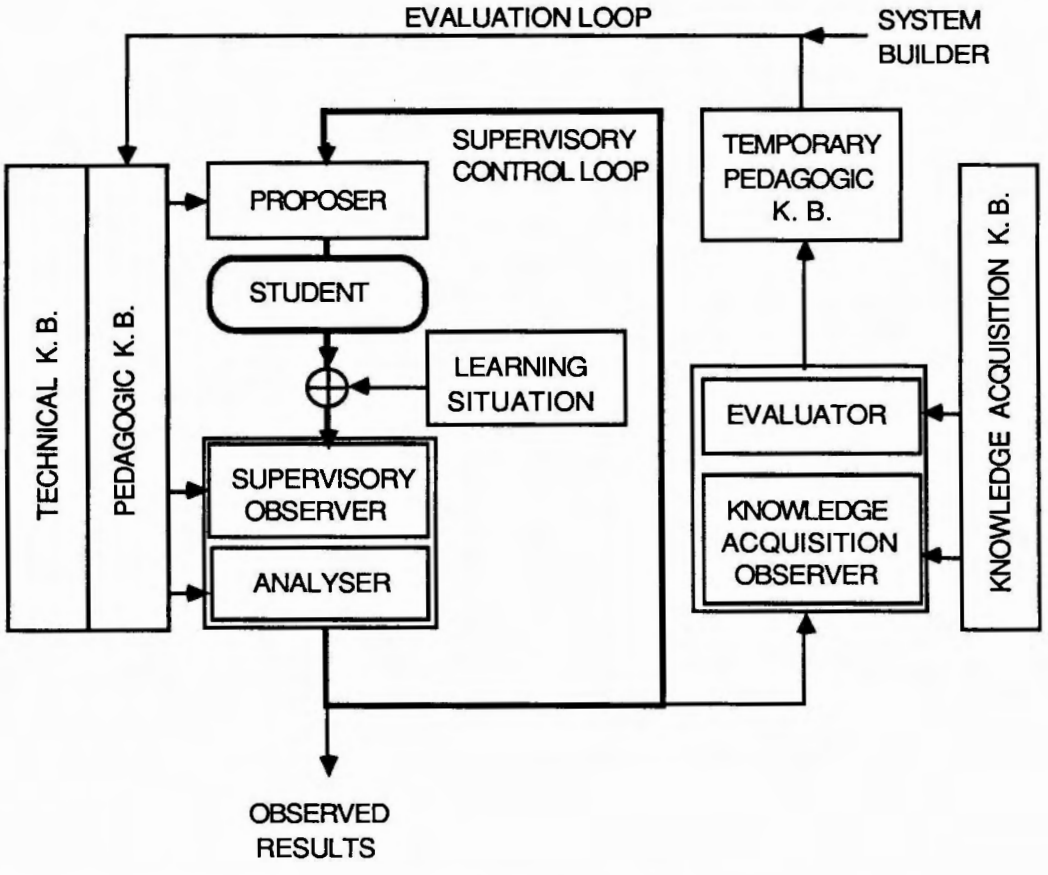


Figure 1 : Pedagogic Model.

Short Term Supervisory Control Loop

The first loop of the pedagogic model concerns the supervisory control of the human learning process. The student is assumed to be faced with a learning situation. He has to try actions or to answer questions from the ITS. Thus, a situation can be generated by (1) the environment, including questions, menus and actions to be taken during training simulations, and (2) the pedagogic knowledge involved to create such a situation. A real situation is defined as a set of facts of the world related by logical operators. For instance, a real situation is to learn how to use the flight management system on an aircraft. Such a situation is very difficult to perceive by a human being when he is not well trained (which is the case in a training context). Thus, the student has to build a set of *situation patterns* (Boy, 1987) to make him able to perceive the real situation as it is for better understanding. The human information processor has been shown to behave using two kinds of processing: (a) the situation recognition which is based on a pattern matching between the real situation and situation patterns already learned, and (b) the analytical reasoning which involves inference. The goal is then to facilitate the student's acquisition of "good" situation patterns.

The same applies to the ITS perceiving what the student is doing in a given situation. This task is alloted to the supervisory control observer module. The observed behavior is a projection of the

real situation, including what the student is doing, on a grid called observation grid (Figure 2). This observation grid will be described later. The observed situation is an input of the analyser. The analyser is a processor using the actual pedagic and technical KB. Output of the analyzer module are sequentially processed by a proposer module which provides a proposition, i.e., either advice or a new pedagogic situation to act on. A pedagogic situation is an exercise designed to help correct previous misunderstanding and lack of knowledge. Generally, a pedagogic situation must be simple enough to be solvable by the student.

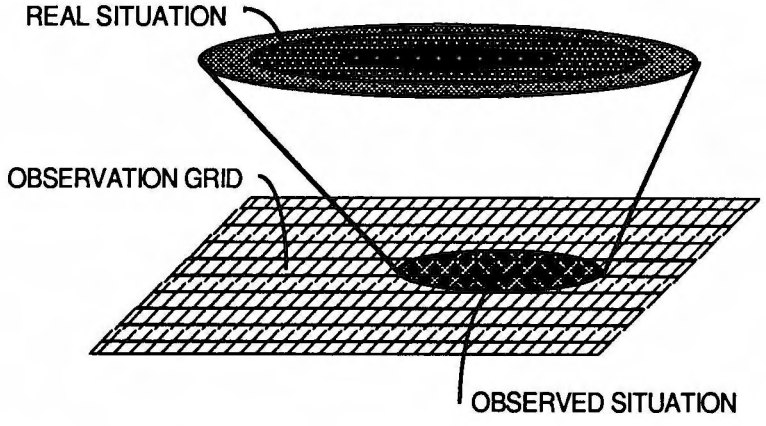


Figure 2 : Observation Grid

Long Term Evaluation Loop

The second loop of the pedagogic model is more specifically devoted to machine learning. The principle is almost the same as for the previous loop. The role of the learning-oriented observer module is to detect new situations for later analysis and possible machine learning. This observer includes the same set of patterns as the supervisory control observer. It is working on those patterns as a partial pattern-matcher, i.e., using analogy, and detecting new patterns by complementing the actual pattern base. Outputs of the observer module are processed by the evaluator module. The evaluator is a processor using an external knowledge base and the "intelligence" of the system builder. This knowledge is methodological. Outputs of the evaluator are modifications of the existing pedagogic knowledge bases.

This loop is not tractable before implementation and testing of the first one. At present, this loop is open and directly regulated by the system builder, so it will not be described here.

3. Observer Model

In the following, a general observer model will be described for the short term supervisory control loop. However, this model can be used for the long term evaluation loop. The observer procedure works using *objects* to be filled in by events coming from the environment. An object is associated with each action of the student. A student action could be an answer or a selection. The behavior inducing each student's action is determined by procedural attachment. Thus, each action either generates a new object or modifies the content of existing objects.

3.1. Observer Knowledge Base.

The goal is to assess the performance of the student. The corresponding knowledge is a set of students' actions. The observer KB or observation grid is modelled using an object-oriented representation. In particular, each student's action is perceived as an object.

The generic object template is the following :

Object associated to Action_A :

Classes :	Exercice_N	
	Problem_Type_M	
Properties :		
	Action Description :	Question_Expression
		Input_Device [mouse, keyboard, ...]
		Effective_Action
	Previous_Attempts :	Number
		{Effective_Action(s)}
	Truth_Value :	[True, False, Not_Known]
	Action_Type :	[Suggested, Volontary]
	Elapsed_Time :	From_Beginning_of_Session
		From_Beginning_of_Exercice
		From_Last_Input
	Student_Belief :	[I_know, I_try_to_see, I_dont_know]
	Behavior :	Procedural_Attachment
		-> Try_and_Error
		-> Slip
		-> Processed_by_Logic
		-> Processed_by_Analogy
	Weight :	[0..100]
	Student_Level :	[Beginner, Medium, Advanced]

An example of an objet is the following :

Object associated to action_4 :

Classes :	Exercice_Flight_Plan_Pages	
	Problem_Type_Page_Changing	
Properties :		
	action Description :	Question_Expression :
		Go_from_Flight_Plan_Page_A_to_B
		Input_Device : Keyboard
		Effective_action: Slew_key_down
	Previous_Attempts :	Number : First
	Truth_Value :	False
	Action_Type :	Volontary
	Elapsed_Time :	From_Beginning_of_Session: 9'34
		From_Beginning_of_Exercice: 3'02
		From_Last_Input: 4'
	Student_Belief :	I_know,
	Behavior :	Procedural_Attachment
		-> Processed_by_Analogy with
		Go_from_Init_Page_A_to_B
	Weight :	12
	Student_Level :	Beginner

Interesting behavioral entities are sequences of actions rather than single actions. Automatic detection of such sequences is a really tricky problem. One way to approach this is to elicit *sequence patterns* by experimentation. An action sequence pattern is an object defined as follows :

Action sequence pattern :

Valid_Contexts :	{list of valid contexts}
Action_Sequence :	{list of actions}

In this paper, we will take the following notation :

$$\{ (C_1, C_2, ..., C_n), (A_1, A_2, ..., A_m) \},$$

where C_i is a context and A_j is an action.

Contexts represent the field of application of the corresponding actions. Contexts and actions, i.e., patterns, are built incrementally after compiling several experiments with students.

3.2. Observer Processor

The observer processor works as presented in figure 3.

Pattern Matching.

After each student action, the processor filters this action through a "relevant" sub-set of action sequence patterns. Such a relevant sub-set is preselected from the whole knowledge base according to the present context. Retrieval criteria will not be described here. The context filter works taking into account the actual situation. It selects a sub-set of patterns called *active* action sequence patterns. Contexts are predefined by the system builder taking into account a priori technical and pedagogic knowledge. In the next research phase, contexts will be updated by the long term evaluation loop.

The pattern matching can be complete or partial. It is complete when every action in the predefined sequence have been performed by the student. Given the action sequence $(a_1, a_2, ..., a_m)$ valid in the context c,

IF $\{a_1=A_1; a_2=A_2; ...; a_m=A_m\}$ and c is member of $\{C_1, C_2, ..., C_n\}$,
THEN the pattern matching is complete.

However, a pattern can be "partially" matched if a reasonably number (taking into account action weights) of "important" actions have been completed. In all cases, several patterns could be matched. Given the action sequence $(a_1, a_2, ..., a_p)$ valid in the context c',

IF $\{a_j=A_{s(j)}$ / where j is member of $\{1, 2, ..., p\}$ $\}$,
where $p < m$
and s is an application from $\{1, 2, ..., p\}$ into $\{1, 2, ..., m\}$,
AND c' is member of $\{C_1, C_2, ..., C_n\}$,
AND $\{ \prod (w(a_j)) \geq T$ / where j is such as $a_j=A_{s(j)}$ $\}$,
where $\prod$ is a conjunctive formula to be defined,
and w is a weight function from $(a_1, a_2, ..., a_p)$ into [0, 1]
and T is a threshold,
THEN partial pattern matching between $(a_1, a_2, ..., a_p)$ and the action sequence pattern $(A_1, A_2, ..., A_m)$.

Conflict Resolution.

Conflict resolution is the process which will select one or several patterns out of the matched ones. Rules for such process is very domain dependant. Rules can select either one hypothesis taking into account a given criterion, or several hypotheses for later parallel processing. Present implementation is based on a single hypothesis selection. For each completely or partially matched sequence, weight factors are combined through a conjunctive formula such as defined above for partial pattern matching. Results of such conjunctive formulae applied to each matched pattern are compared together and the best is then selected.

Inference.

When an action sequence pattern is selected, a relevant hypothesis is suggested to the diagnoser. As this system is connected with a dynamic environment, i.e., erratic actions of the student, patterns can be fired even when the diagnoser is working. Such characteristics impose on the diagnoser the necessity to be able to revise its beliefs on facts it has already deduced. In other terms, if the level of the student has been deduced as "advanced" and an action sequence pattern suggests that his/her level should be "beginner", then a belief revision must occur. Thus, the diagnoser includes two parts: a problem solver and a belief maintenance system. The role of the problem solver is to deduce facts from initial facts and rules in order to prove a selected relevant hypothesis. The belief maintenance system can revise truth and values of facts according to a new situation. This process is performed by resetting rules already used. Such a feature allows incremental reasoning in changing environments.

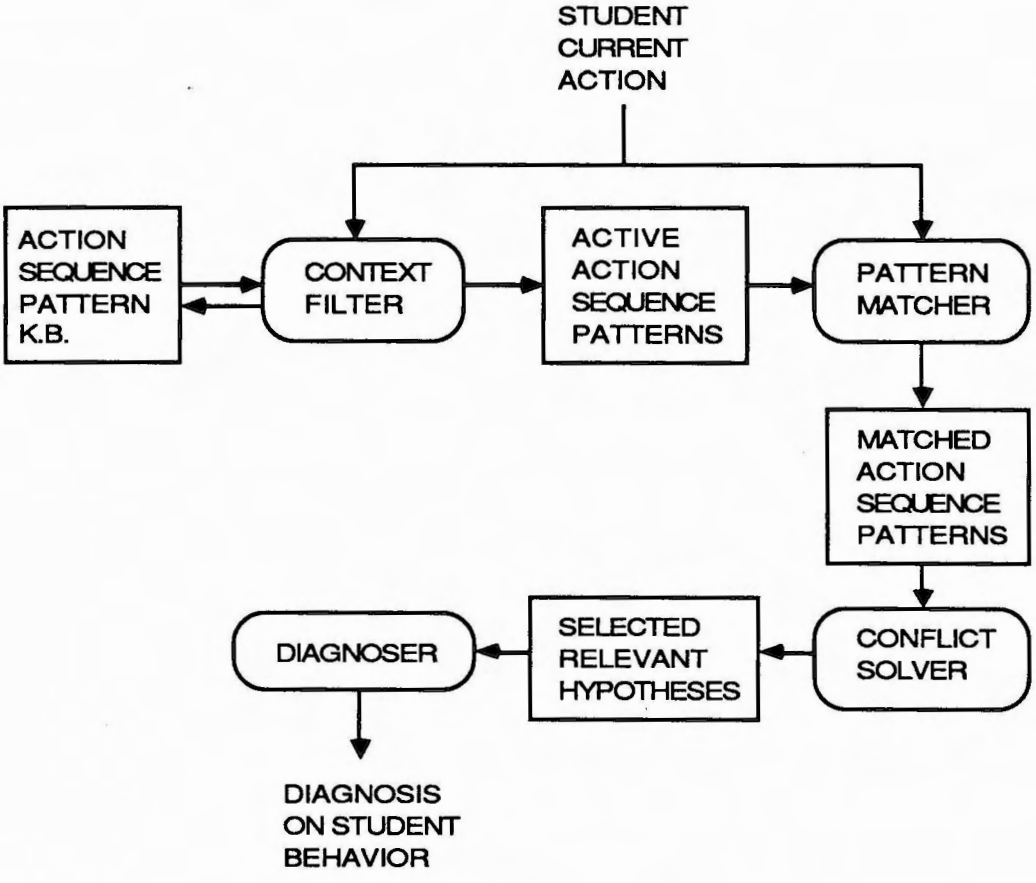


Figure 3 . Observer Processor.

4. Knowledge Acquisition Process

In our approach, the main difficulty in building the system is to design *situation patterns*, i.e., action sequence patterns, and chunks of *analytical knowledge*, i.e., diagnoser rules. Such knowledge will be acquired using an adapted acquisition procedure. At present, this procedure is performed manually. However, it will be described formally in the following.

4.1. Classical methods

There is no universal pedagogic knowledge. The difficulty in such knowledge acquisition (KA) is the variability of sources. Moreover, a given pedagogy can be very efficient with one student and very bad with another. Thus, student reactions to pedagogic inputs must be observed and

processed in order to correct initial pedagogic strategies.

The first approach in the KA process is to enter into the pedagogic knowledge base a presumably "good" sub-optimal knowledge. This first approach includes the following classical methods of manual KA (Kidd & Welbank, 1984): interviews of tutors (tape-recorded, structured (Lafrance, 1986), or using interview tools (Boose & Bradshaw, 1986)), brainwriting (Warfield, 1971), brainstorming, direct observation (Boy, 1986), and verbal protocol analysis (Ericsson & Simon, 1980).

The second approach can be characterized by building *knowledge design interfaces* (Rappaport, 1986). Such interfaces are accessible and user-friendly to the tutor him/herself. They are designed for the tutor to enter his/her own pedagogic knowledge according to each significant observed situation. This approach reduces bias introduced by a knowledge engineer. For example, good tutoring systems have been shown to be built by computer scientists themselves for computer science teaching. The only two systems which are commercially available and in every day use are in the domain of computer programming: PROUST (Johnson & Soloway, 1985) and LISP tutor (Anderson & Reiser, 1985).

The last approach is machine learning (Michalski et al., 1983, 1986). It transforms an initial sub-optimal knowledge base into one which is both more general and better adapted to a particular student. Machine learning starts to be an interesting approach when initial knowledge bases are reasonably big, i.e., we learn very little when we know very little, and the converse. Machine learning is presently divided into two approaches: similarity-based learning (SBL) and explanation-based learning (EBL) (Kodratoff, 1986).

4.2. Building patterns

At the beginning of ITS construction, the KB is sub-optimal. Each situation pattern, for instance, is simple and relies on a complex analytical KB. For example, a delay on the reaction time of the student is an a priori situation pattern which implies a complex analysis to infer. Either he/she does not understand a system question or the overall situation itself. As the ITS is used, the system builder (who can be a tutor) becomes more knowledgeable about accurate and relevant observation patterns. Pattern construction is a top-down approach, i.e., patterns are designed from system-to-be-learnt specifications, and a bottom-up approach by experimentation, i.e., by observing student behaviors and reactions when faced with the ITS.

4.2.1. Aim of the Method

Learning can be seen as a two-directional transformation: specialization and generalization. Most of efforts in machine learning research concentrate on generalization, however as far as expertise is concerned specialization plays a primary role. Indeed, an expert is not necessarily "intelligent" in the sense of highly general sophisticated knowledge, but he has very domain-specific specialized patterns built from experience. Generalization consists of building disjunctive chunks, as opposed to specialization which consists of building conjunctive chunks.

The knowledge acquisition method presented here belongs to the class of EBL techniques. In the present method, explanation comes from experts as well as from observers. Input data to the learning process is procedural knowledge. The knowledge acquisition process consists of three phases: (1) the gathering of procedural knowledge for construction of an initial sub-optimal KB, (2) the transfer (specialization) of elements of this analytical knowledge into situational patterns by experimentation, and (3) the structuration of particular chunks of the analytical KB. To a certain extent, this method is similar to the chunking method developed in SOAR (Laird, Newell, & Rosenbloom, 1984). The difference is that here chunking effectively transfers chunks of knowledge from the AK base to the situational knowledge base. Situation pattern-oriented chunks permit parallel processing not only of the situational and analytical knowledge bases, but also several analytical reasonings simultaneously. As in SOAR, the search in the AK base is reduced because of the specialization process.

4.2.2. Knowledge Acquisition by Specialization/Structuration

Initial Sub-Optimal KB.

The first phase of the process consists of building a sub-optimal KB which will be refined by experimentation. The sub-optimal KB includes "small" rough situation patterns which are parameter-sensitive "demons" suggesting further analytical reasoning. The initial analytical KB describes how the system to be controlled works. It is split into a few sub-sets of rules, each of which includes large numbers of rules (Figure 4).

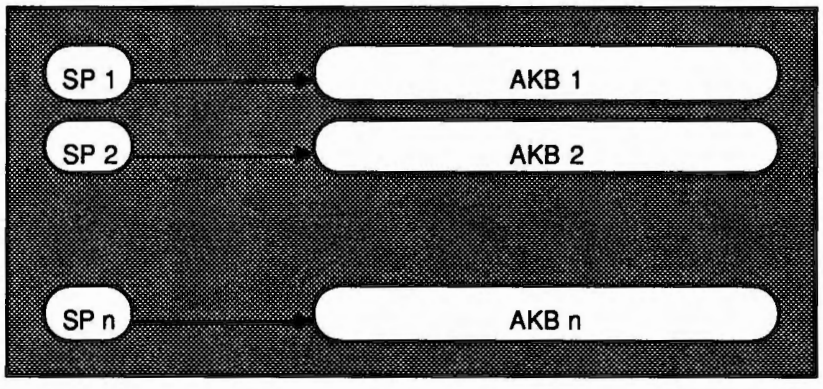


Figure 4 : Initial Sub-Optimal Knowledge Base
(SP: situation pattern; AKB: analytical knowlegde base)

In our ITS project, an example of the original sub-optimal KB is:

SITUATION PATTERN:

IF { Time_spent > 45 sec. } THEN **suggest** Hypothesis(A)
AND start-analytical-reasoning.

ANALYTICAL REASONING:

IF Pedagigic_KB_Validation THEN Hypothesis(A)
AND **reset** Pedagigic_KB_Validation.

This preliminary "analytical" rule works in backward chaining. The subsequent sub-set of rules is used generally in forward chaining and can be summerized in the following graph (Figure 5).

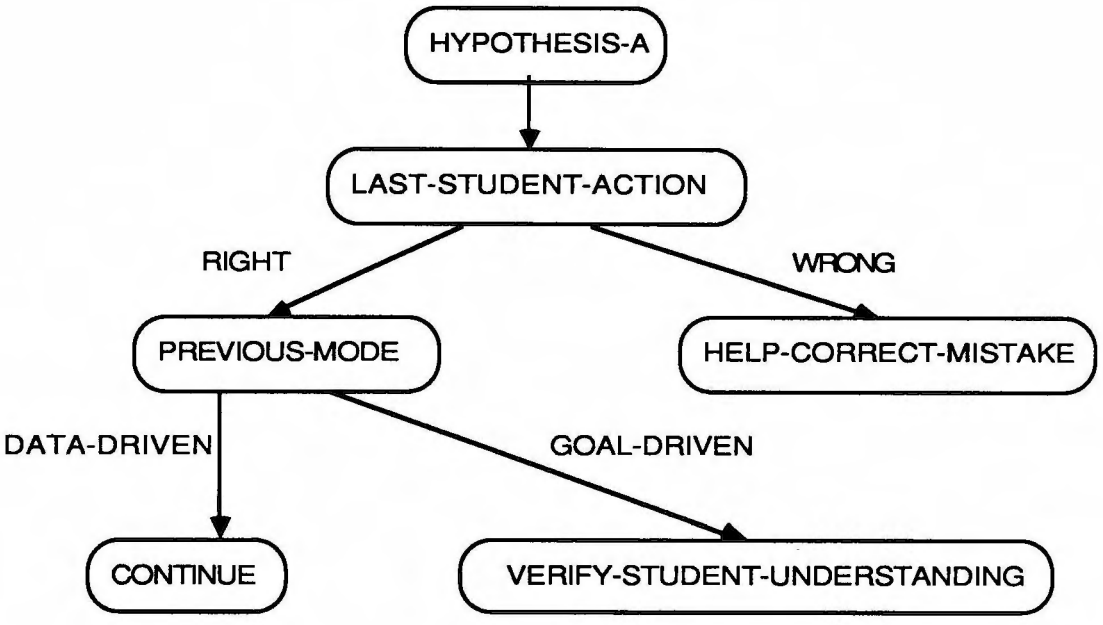


Figure 5 : Analytical Search Tree

Specialization.

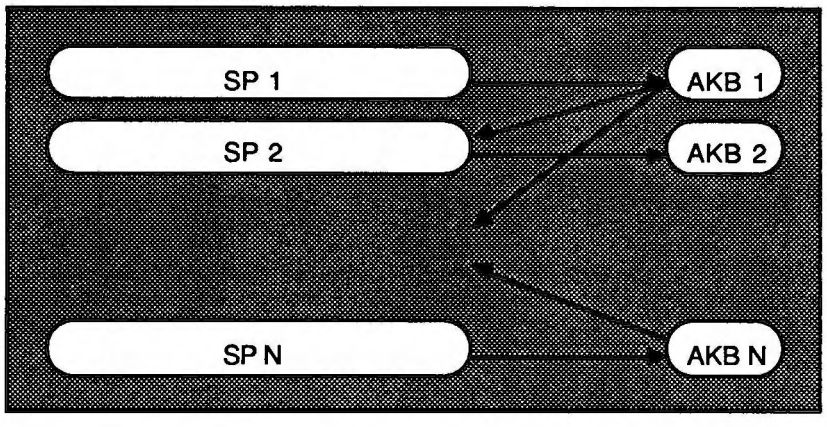


Figure 6 : Acquisition of Specialized Patterns

It has been shown that as professors gain experience in teaching, their situation patterns become more complex (Figure 6), dynamic and numerous (Boy, 1987). In the example described above, it happened that all experts considered the node PREVIOUS_MODE in the early stages of the detection process. In this example, PREVIOUS_MODE could be either DATA_DRIVEN or GAOL_DRIVEN. Thus, this element of knowledge had to be tranferred to the situational part of the KB. The specialisation of the KB has been carried out as follows.

(1) Detection of the attributes or instances of the node to be transferred. Here, PREVIOUS_MODE can be DATA_DRIVEN or GOAL_DRIVEN.

(2) Identification of the initial situation pattern leading to this node and building a number "n" of new patterns corresponding to the number of values or instances of the transferred node. Each new pattern is a conjunction of the content of the initial pattern and each value or instance of the transferred node. In the above example, the new situation pattern is:

IF { Time_spent > 45 sec.
AND PREVIOUS_MODE (DATA_DRIVEN) }
THEN **suggest** Hypothesis(A1)
AND start-analytical-reasoning.

IF { Time_spent > 45 sec.
AND PREVIOUS_MOVE (GOAL_DRIVEN) }
THEN **suggest** Hypothesis(A2)
AND start-analytical-reasoning.

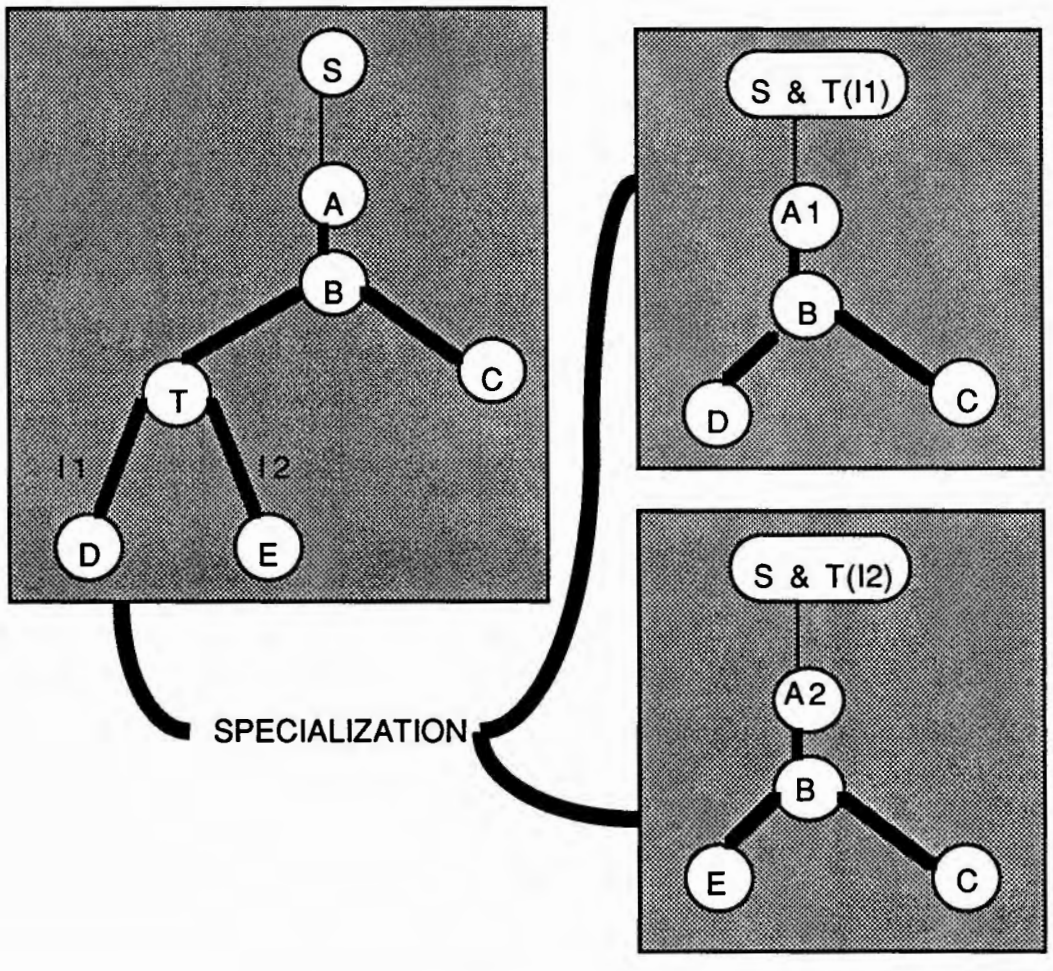


Figure 7 : Specialization Process.

(3) Restructuring of the analytical KB. As n patterns have been created in step (2), the analytical KB must be split into n corresponding sub-sets. This transformation leads to duplication of the part of the analytical KB up to the transferred node and keeping the n branches corresponding to the instances of the transferred node. In a more schematic form, let T be the node to be transferred, I1 and I2 are particular values or instances of T, S is the initial situation pattern, and A,B,C,D,E are other nodes. The specialization process is given in Figure 7.

Structuration.

The structuration process includes both analytic (explanation-based) learning on the analytical KB side and heuristic (similarity-based) learning on the situation pattern side.

Step (3) in the specialization process stresses the need to structure the part of the graph up to the transfer node. The sub-set corresponding to Hypothesis(A) will be called A. Structuration in the analytic KB introduces the concept of hierarchy among the sub-sets of rules. In the above example, sub-sets A1 and A2 have been built from the initial sub-set A. A1 and A2 both include the same sub-set of nodes (B and C). Thus, these nodes are general to the disjunct (A1 v A2). It follows that structuration leads to the building of disjunctions of sub-sets of rules. The resulting structure of the analytical reasoning will be as follows. When a situation pattern is fired, an hypothesis is suggested. To prove this hypothesis, a general sub-sets of rules has to be activated. If these rules are satisfied, the reasoning continues in forward chaining.

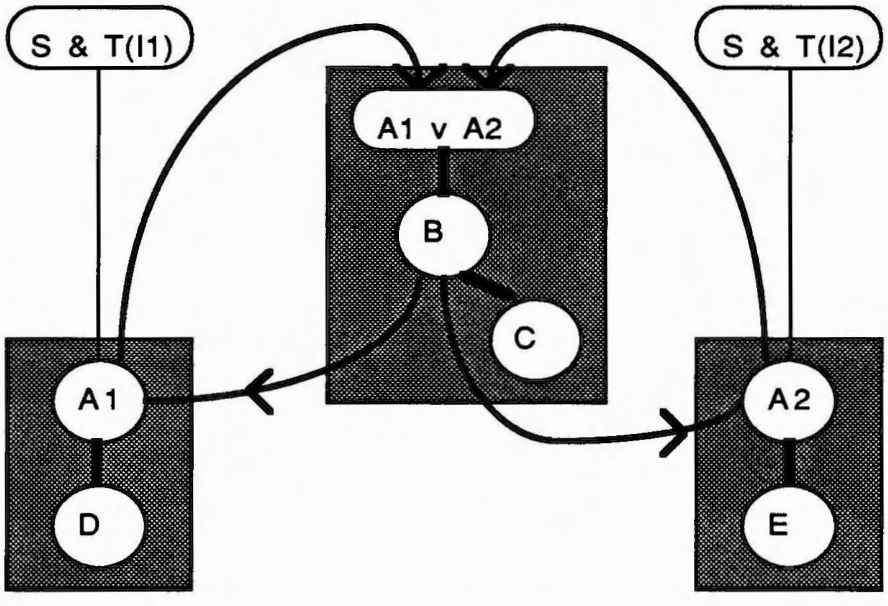


Figure 8 : Structuration Process

The implementation of this technique in the form of rules leads to the first backward chaining rule (for A1 for instance):

IF Pedagogic_KB_Validation THEN Hypothesis(A1)
AND **reset** Pedagogic_KB_Validation
AND **suggest** Hypothesis(A1vA2).

and the sub-set (A1 v A2) starts with a backward chaining rule of the same format. If a first backward chaining rule does not include a suggested action in its right hand-side, the corresponding sub-set of rules will be called a terminal sub-set. Otherwise, it will be a common sub-set. Each sub-set of rules has a given priority according to its degree of generality. This feature permits recursivity on several general sub-sets of rules. Note that a general sub-set of rules can be initiated directly by its own situation pattern, if there is one.

5. Discussion

Our first intention in this work was to develop a general theory for knowledge acquisition. The present attempt is a part of a wider research effort on operator assistant systems. Thus, we tried to tackle intelligent tutoring in this framework. In particular, the Situation Recognition Analytical Reasoning (SRAR) model has been developed (Boy, 1988b) and is being used as a representation reference for implementing student behavior observables. Indeed, observation is the key factor in this approach. As shown above, action sequence patterns are the limiting entities to be acquired by experimentation.

We are presently developing a flight management trainer which will have the functions described above. We are using PLATO author language for implementing various pedagogic situations, e.g., a frame display is given in figure 9. AIRBUS trainees are the subjects for the incremental knowledge acquisition. At this stage, a first ITS mockup has been developed an an IBM PC. This includes a technical KB and a sub-optimal pedagogic KB. The observer module is being developed using NEXPERT *Object*.

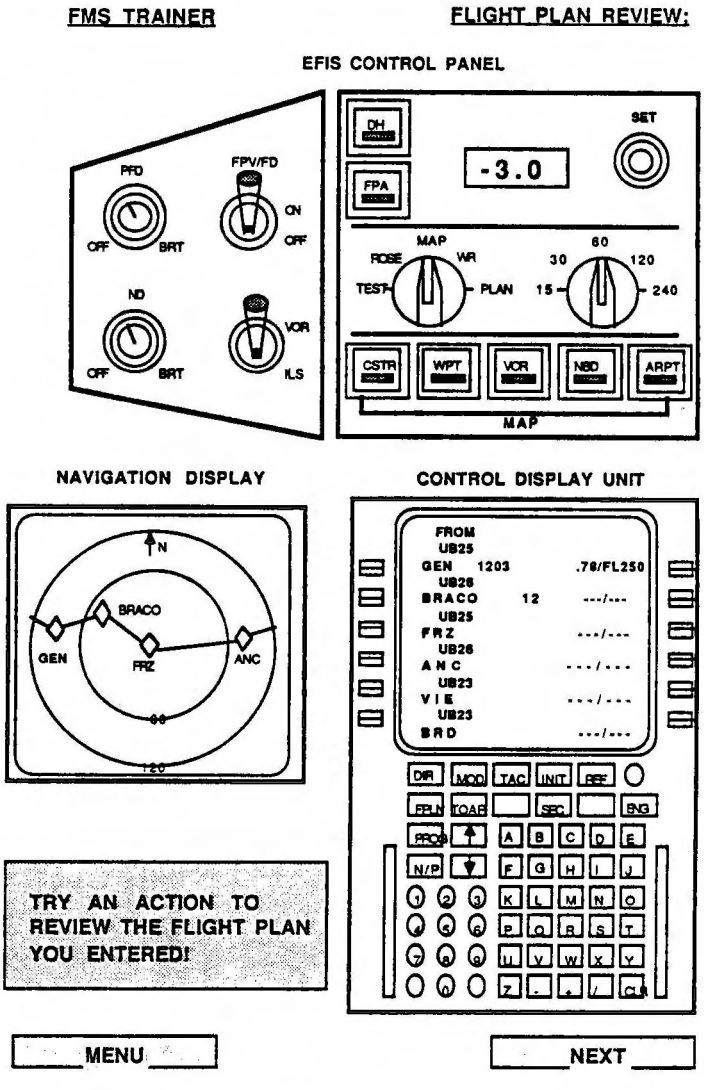


Figure 9: Example of interactive frame display.

Acknowledgements:

This work was supported by AIRBUS INDUSTRIE contract n° 304 020.

Special thanks to P. Medous, F. Hernando and AEROFORMATION intructors for their excellent comments and advice. Thanks to AIRBUS INDUSTRIE test pilots and FMS experts for providing us detailed knowledge on FMS. Thanks to Philippa Gander and Alain Garès for critical comments on the manuscript.

References

Aderson, J.R., & Reiser, B.J., (1985). The LISP Tutor. *Byte*, 10 (4).

Boose, J.H., Bradshaw, J.M., (1986). Expertise Transfert and Complex Problem Using AQUINAS as a Knowledge Acquisition Workbench for Expert SYstems. *Proceedings of Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, November.

Boy, G.A., (1986), An Expert System for Fault Diagnosis in Orbital Refueling Operations. *AIAA 24th Aerospace Science Meeting*, Reno, Nevada, November.

Boy, G.A., (1988a). Operator Assistant System. *International Journal of Man Machine Studies*. To Appear.

Boy, G.A., (1988b). Human-Machine Distributed Intelligence: The Operator Assistant Concept. *Expert Systems and Their Applications*, Avignon, June.

Ericsson, K.A., & Simon, H.A., (1980). Verbal Report Data. *Psychological Review*, vol. 87, May.

Feigenbaum, E.A., Barr, A., (1982). *The Handbook of Artificial Intelligence*. vol. 1, vol. 2.

Forcheri, P., Molfino, M.T., (1986). A Historical Approach to Educational Computer-Based Systems. *Revue d'Intelligence Artificielle*.

Freiling, M., Alexander, J., Messick, S., Rehfuss, S., and Shulman, S., (1985). Starting a Knowledge Engineering Project: A Step-by-step approach. *AI Magazine* 6(3): 150-163.

Friedland, P., (1981). Acquisition of Procedural Knowledge from Domain Experts. *Proceedings of Seventh International Joint Conference in AI*, pp. 856-861.

Gevarter, W.B., (1984). Artificial Intelligence Computer Vision and Natural Language Processing. Park Ridge, N.J.: Noyes.

Johnson, W.L., & Soloway, E., (1985). PROUST. *Byte*, 10 (4).

Kidd, A., & Welbank, M., (1984). Knowledge Acquisition. *State of the Art Report on Expert Systems*, J. Fox Ed.

Kodratoff, Y., (1986). *Leçons d'Apprentissage Symbolique Automatique*. Cepadues Edition.

Lafrance, M., (1986). The knowledge Acquisition Grid: A Method for Training Knowledge Engineers. *Proceedings of Knowledge Acquisition for Knowledge-Bsed Systems Workshop*, Banff, November.

Michalski, R.S., Carbonell, J.G., Mitchell, T.M., (1983). *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto.

Mitchell, T.M., Carbonell, J.G., Michalski, R.S., (1986). *Machine Learning: A Guide to Recent Research*, Kluwer Academic Publisher.

Powell, D.R., & Andrews, A.E., (1987). Design of an Intelligent Tutoring System for Aircraft Recognition. *IEEE Control System Magazine*, Vol. 7, No.3, June.

Quinlan, J.R., (1984). Fundamentals of the Knowledge Engineering problem. In *Introductory Readings in Expert Systems*, Ed. D. Michie, 33-46. London: Gordon and Breach.

Rappaport, A., (1986). Multiple Subspaces for Problem Solving in the Process of Knowledge Acquisition. *Proceedings of the Human-Machine Interaction and Artificial Intelligence in Aeronautics and Space Workshop*, G. Boy Ed., CERT, Toulouse.

Sleeman, D., & Brown, J.S., (1982). *Intelligent Tutoring Systems*. Academic Press, London.

Warfield, J.N., (1971). *Societal Systems: Planning, Policy and Complexity*. John Wiley & Sons.

Shared Causal Knowledge as a Basis for Communication between Expert and Knowledge Acquisition System

John Bradshaw
Computer Laboratory, University of Cambridge, Corn Exchange Street, Cambridge.

Richard Young
Applied Psychology Unit, Medical Research Council,15, Chaucer Road, Cambridge.

Extended Abstract

It has often been said that knowledge acquisition is the 'bottleneck' in the design and implementation of knowledge based systems [eg. Mor87, Gai87]. It is our view that the problems of knowledge acquisition are a combination of two problems - that of knowledge representation and knowledge elicitation. We make a fine distinction between acquisition and of elicitation: acquisition deals with the whole '*from expert to knowledge base*' process where as elicitation only deals with the '*from expert*' section.

We accept that there are problems specifically to do with the elicitation process but contend that this process is not helped by the knowledge representation techniques currently available. It would seem that for some domains, where the problems are well understood and where the means of arriving at a solution can be clearly articulated, the establishment of a knowledge based system is rather simpler than for other domains where the problem and the solution are less clearly understood. The reason for this ability to handle a few domains well is principally because these domains can be clearly articulated so it has been relatively easy to develop suitable representations to accommodate them.

Our aim is to produce an automated system which, because it understands the structure of the knowledge base will be able to intelligently assist in the acquisition process. The type of knowledge which will be useful in the elicitation phase is the same knowledge which a 'second generation' expert system requires over and above that currently captured by 'first generation' systems. We are looking to develop a technique which will enable a system to initially acquire/build general domain models and later add specific knowledge to produce knowledge bases for specific problems. The system should be able to

build on knowledge it already has by using that knowledge to communicate with the expert - communication based on shared knowledge. A key feature that would be required of the knowledge base in such a system is flexibility and it is our contention that the representation strategy outlined here is more flexible than currently popular techniques.

In this paper we shall describe recent work towards the establishment of a representation strategy which allows the same knowledge to be used in a number of different ways. We also show how general domain knowledge can be captured just once and then combined with different sets of problem specific knowledge to solve different problems in the same domain. We feel that the description which we present here will go some way to easing the bottleneck on the knowledge base side of the acquisition process. It is also our contention that because the knowledge base will then be more powerful, a more meaningful dialog will be possible between expert and acquisition system based on the notion of shared knowledge.

Functional Description of Causal Systems

It would be extremely beneficial if a causal system could be represented at different levels by 'components' which all have the same structure. This would provide a convenient way of describing and decomposing (sub)systems for the purposes of diagnosis, simulation and other tasks. It would also enable the acquisition system to 'understand' the knowledge already collected, by virtue of its knowledge of the structure, and thus enable it on the basis of this shared knowledge to communicate with the expert in a more informed way. We have therefor developed the idea of a 'virtual component' and described its characteristics and behaviour.

A virtual component is any part of a system which one may choose to describe. It can be a description of the a whole system, a subsystem or a single object. We take the 'black box' view of a virtual component. All virtual components receive control and/or material input and produce behaviour and possibly material output. Material need not be material in any tangible sense but should be regarded as any input which contributes directly to the formation of the the output. Thus data can be material input if it is manipulated to form the (material) output - new information. Control and material input are

distinguished by the following criteria: Material input is input which is used in the formation of material output, control input is used to determine if material should be produced or not.

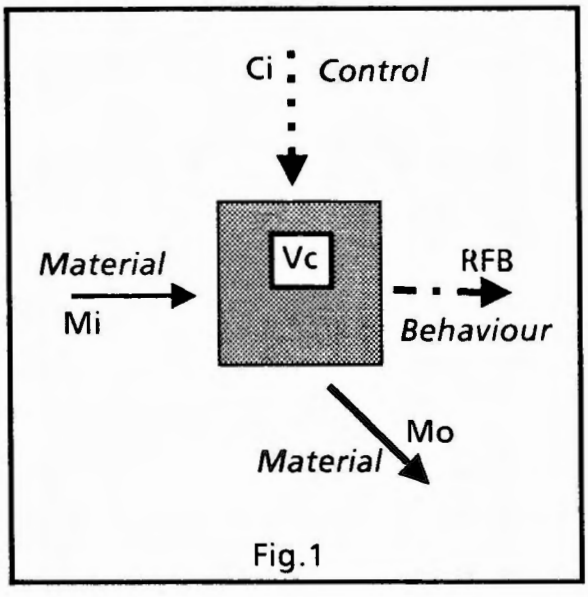


Fig.1

A virtual component (Vc) always produces RFB (Resultant Functional Behaviour) what ever the state of control input (Ci) and material input (Mi). Material output(Mo) is only produced when Ci allows it and when Mi is ***available***. We say that a Vc's RFB is ***energetic*** if the Vc is producing Mo; RFB is ***alert*** if Mi is ***available*** but is no Mo is being produced and otherwise RFB is ***quiet***. We can also say that if Ci is in a state in which it will not allow Vc to produce its Mo then Ci is ***passive***, otherwise it is ***active***.

If there is no Ci (as opposed to Ci being passive) then Vc's RFB is quiet if Mi is not-available, RFB is energetic if Mi is available, Vc' RFB no longer has the capacity to be in an alert state. In other words *no Ci is equivalent to a Ci which is always active*.

The three following are therefor the only legal combinations:

Mi	**Ci**	**Mo**		**RFB**
not-available		not-available	=	quiet
available	passive	not-available	=	alert
available	active	available	=	energetic

We have generalised this notion of a component to allow a user to describe a standard type of component and to record specific instances of this type. For example a switch is a type of component which takes some material input and gives out the same material output. A timer, a tap and a thermostat are all instances of a switch. They are all different in their material and in their manner of control. The system can be told that switches have the same (material) output as input. In the case of the timer the material input is electricity. If electricity is available then the timer's Mi is available and if its Ci is active then it will output electricity. We have a means here of providing a general description of a type of component and combining this later with individual descriptions of specific instances of this type.

Each component in a system has associated with it five values/states: Mi, Ci, Mo and Rfb (as described above) and its actual value - often on or off, or numeric (as for temperatures) but can be defined by the expert as anything, eg. hot or cold as for a heater. These values are broadcast for all components to see and use. The destination of these results are of no interest or concern of the source component - when a pump pumps water it does so with no regard for the use or destination of the water, its task is simply to move the water on. The same approach is taken here. The implications of the component's results are the concern of the wider system and of the users of the results. It is those components which use the results who give the results meaning.

This representation strategy allows the expert to describe his system in a very top down fashion, returning later to fill in the detail and perhaps to correct simplifications which where made to help with the initial description - a natural way of teaching. A boiler could initially be described as a single unit before showing the heater and thermostat components. Also, having described a boiler which uses electricity it is a short step to converting it to gas. We are able therefor to minimise the time, and effort, expended on redescribing the same system again. Given this ability we contend that a knowledge elicitation process need not, indeed should not, start from scratch for every new knowledge base ordered. If the next knowledge base is similar to one already produced then the common parts can be transferred directly and the elicitation process can

concentrate on the points of departure between the new system and that which is already understood and represented.

Conclusion

We believe that through the representation strategy outlined above we will be able to make the acquisition task easier because the target representation is flexible enough to handle different kinds of domain and knowledge and to enable this knowledge to be used in different ways - not least of which is in the acquisition process its self through a notion of communicating using shared knowledge. It is our expectation that this knowledge representation strategy should allow the expert to produce systems which could incorporate other, previously developed, knowledge bases and which will allow him to start from a simple, perhaps inaccurate, general description of the system and move towards a fuller more accurate description.

We have developed an alternative knowledge representation strategy because we felt that it was important to bring together the design of the knowledge base structure and the acquisition process. We feel that we have a design here which has some potential in easing some of the problems which are currently encountered in the acquisition process. Some writers, eg [Gai87], seem satisfied in continuing to use pure rule based systems but Morik [Mor88] has shown that more information is required if more powerful acquisition systems are to be built. Indeed her conclusion, in which she sums up why modeling is the central issue in both knowledge acquisition and machine learning could just as well have been written to support our view of knowledge representation design with respect to knowledge acquisition.

"- Modeling is an interactive process, so the system should be interactive at all points in the modeling process

- Models can be operational, thus the system should not only store the model but also answer questions about the knowledge"

References

[Gai87] Gaines, B., An overview of Knowledge acquisition and transfer, Int. J. Man-Machine Studies, **26**, pp 453-372, 1987.

[[Mor87] Morik, K., Knowledge acquisition and Machine learning - the issue of modeling, in Proc. of First European Workshop on Knowledge Acquisition for Knowledge Based Systems, Reading University, IEEE, 2-3 Sept., 1987.

[Mor88] Morik, K., Sloppy Modeling, to appear in Knowledge Representation and Organization in Machine Learning (ed Morik, K.), Lecture notes, Springer, 1988.

DESIGNING INTERACTIVE EXPERT CLASSIFICATION SYSTEMS THAT ACQUIRE EXPENSIVE INFORMATION "OPTIMALLY"

Louis Anthony Cox, Jr., Ph.D.

U S WEST Advanced Technologies

6200 South Quebec Street
Englewoood, CO, 80111

ABSTRACT

A fundamental challenge in designing interactive knowledge acquisition and modeling systems concerns the order in which the system should ask questions in seeking information that will help it to answer a user's query. This paper examines the problem of "optimal" knowledge acquisition through questioning in contexts where it is expensive to obtain the answers to questions. An abstract model of an expert classification system -- considered as a set of logical classification rules supplemented by some statistical knowledge about attribute frequencies -- is developed and applied to analyze the complexity of and to present constructive algorithms for doing probabilistic question-based classification. This model corresponds to a simple but important class of expert systems that has been widely used in fault prediction, fault diagnosis, and process control applications. We present a new class of double sort algorithms for generating and scheduling questions in such systems during a knowledge acquisition dialogue, and contrast this approach to eliciting relevant knowledge with the more usual "recursive partitioning" approaches (e.g., ID3 and its extensions) now being used in artificial intelligence and statistics to learn classification rules from empirical data.

This paper describes how to use a knowledge base to efficiently classify objects in situations where each object's class depends only on the values of its attributes. It is assumed that prior statistical information about attribute frequencies supplements the logical classification rules in the knowledge base, and that attribute values for an object can be observed for some cost before drawing final conclusions about the object's class. This work is directed toward the design of practical interactive expert classification systems for contexts where acquisition of information is expensive -- e.g., because it requires costly experiments or diagnostic tests. The trade-offs involved in "optimal" rule-based classification are treated in a standard statistical decision theory framework. It is assumed that the classification rules are already known (e.g., through application of a rule-learning algorithm like ID3 to a large training set) but that the order in which to apply them during a dialogue with the user needs to be (adaptively) determined. This framework leads to a new class of heuristic algorithms that has some computational and performance advantages over the class of "recursive partitioniong" (or "tree structured") heuristic algorithms currently being used in both statistics and artificial intelligence to learn classification strategies from empirical data.

1. Formulation of the Minimum Expected-Cost Classification Problem

The problem of efficient classification of an object or "case" based on sequential inspection of its attributes can be approached from either an artificial intelligence (A.I.) or a traditional statistical decision theory/ pattern recognition perspective. An A.I. formulation is as follows. Suppose we have a set of logical rules for classifying cases based on the values of each of n descriptive attributes. Such rules map sets of attribute values into corresponding assigned classes. In the simplest situation, to which a substantial fraction of current research on machine learning is devoted, all attributes are binary: each one is either present or absent in an object (14-16). Binary attributes will be assumed throughout this paper. If the set of rules is deterministic, then it can be expressed in the following canonical form:

```
C1 if [a11 & a12 & ... & a1n]
C2 if [a21 & a22 & ... & a2n]                    (1)
.
.
.
Cm if [am1 & am2 & ... & amn]
```

This array is called a rule base, or an expert classification knowledge base. Each of the aij's is a proposition stating either that attribute j is present (aij = 1) or that it is absent (aij = 0) or that it does not matter for the classification conclusion expressed on the left hand side of rule i whether attribute j is present (aij = ?). (In the latter case, the corresponding aij automatically evaluates to "true." Such "don't care" conditions can be omitted, but it is useful for uniformity of notation to include them.) The propositions aij have truth values for any object that is being classified. Nondeterministic rules of the form

(Cj with confidence factor p) if [ai1 & ai2 & ... & ain]

can be included within this formalism, if desired, by treating the value of one or more attributes j as unobservable, so that only the probability that aij is true is known.

Each row in (1) is a rule (a Horn clause in logic programming.) The right-hand sides in square brackets are called clauses. Each of the labels C1,...,Cm is the name of a class; Ci asserts that that the class of the object is Ci. If all of the propositions in a clause evaluate to "true" for an object then the clause is said to be satisfied and the rule asserts that the object belongs the class named on the left-hand side of the rule. If Ci and Cj are the same, then rule base (1) asserts that, if either of the clauses [ai1 & ai2 & ... & ain] or [aj1 & aj2 & ... & ajn] is satisfied, then the object being classified belongs to class Ci = Cj. [Thus, any logical rule that is expressed in disjunctive normal form (DNF) -- a canonical form that has been widely used in the machine-learning literature (14-16) -- can be reexpressed in the canonical form (1) by distributing disjuncts across rules.] In practical applications, class labels might represent predictions, diagnoses, or presecriptions for control actions to be applied to the object being classified.

Previous work emphasizing induction of logical classification rules from data has often assumed that a set of objects is available for which both the values of each object's attributes and the correct classification of each object are known (1-8.) From the data in this "training set," the induction algorithm seeks to learn a general decision rule, i.e., a classification rule, mapping logical combinations of attribute values into corresponding classes. This general classification rule can then be reduced to a set of Horn clauses as in rule base (1), post-edited to remove redundant or uninformative attributes from the right-hand side clauses, and implemented in an expert system (7). Examples of induction algorithms that have been successfully used to learn classification rules from data in a wide variety of commercial applications include Quinlan's ID3 algorithm for objects characterized by the values of a few discrete attributes (6-8), and the CART system for classification of objects with either discrete or continuous attributes (1-4). An important special case occurs when there are only two possible classes. Then the correct decision rule may be interpreted as a logical concept to be learned, and the learning mechanism or algorithm is trained to classify each object as either an instance (class 1) or as not an instance (class 0) of the target concept. This paper addresses the more general case in which there may be any finite number of classes.

Instead of a training set, this paper assumes that some prior statistical knowledge about attribute frequencies is available to guide classification. Specifically, the prior probability pj that each attribute j will be present in any object is known, and all attributes are assumed to be mutually statistically independent: whether one attribute is present does not change the probability of any other attribute being present. Rather than trying to learn new classification rules from a training set of data, we assume that a set

of logical classification rules having the canonical form (1) is given (perhaps after applying a learning algorithm such as ID3 to a large training set.) The problem addressed here is not how to learn such rules during a training phase, but how to apply them efficiently to classify new cases in application. This is difficult because, in contrast to most previous work on machine learning, we assume that it costs an amount cj to learn the value of attribute j for an object. It is therefore not enough to know the logical rules mapping attribute values into classes. Rules are applied to new objects by asking questions in an effort to show that a right-hand side clause is satisfied, so that a left-hand side conclusion can be drawn. When questions are expensive to answer, a strategy for applying these rules so as to minimize the average cost of reaching a conclusion must be developed. Developing such strategies is the principal goal of this paper.

The purpose of classification is ultimately assumed to be to help decide what action to apply to each object or case being classified. A set A of possible actions is assumed to be available, exactly one of which must be selected for application to each object. Each object has some true but unknown state (e.g., its true class) belonging to a set X of possible states. If action a from A is applied to an object whose true state is x, then a loss L(a,x) results. This conventional statistical decision theory formulation generalizes the "pure" classification problem in which A = X and in which the loss function L(a,x) is L(a,x) = 0 if a = x and L(a,x) = 1 otherwise. Allowing there to be a different number of actions than classes, and concentrating on the mapping from attribute values to actions, substantially extends the range of practical applications described by this model. The "classes" C1 through Cm in (1) should therefore be interpreted henceforth as actions or decisions. Problems with this structure will be called generalized classification problems (GCP's.) Standard example interpretations of this mathematical formulation include (i) X is a set of possible diseases and A is a set of possible prescriptions; (ii) X is a set of possible signals and A is a set of possible responses (e.g., fire a torpedo or don't, in a military application); (iii) X is a set of possible symptoms and A is a set of possible diagnoses. The minimum expected-cost generalized classification problem (MEGCP) is to construct a question-asking and action-selection strategy that balances the costs of inspecting (i.e., discovering) attribute values against the expected losses from misclassification [i.e., from incorrect action) so as to minimize total expected loss, defined as the sum of inspection costs and the "terminal loss" L(a,x).

The following sections examine the computational complexity of MEGCP and propose a new, computationally efficient heuristic algorithm for solving it. The new algorithm is contrasted with recursive partitioning or "tree-structured" algorithms, which provide an alternative heuristic approach to sequential inspection and classification when observation costs can be ignored. The context for the analysis is the problem of designing an interactive expert classification system that will adaptively generate a sequence of questions to minimize the expected cost of reaching a conclusion, including the cost of misclassification error.

2. Formulation as an Expert System Design Problem

Many expert systems can be looked at abstractly as pattern-recognizers or classifiers. The expert system paradigm is roughly as follows. A builder or user supplies the machine with some generic knowledge about an application domain, formulated as a system of logical rules as in the canonical rule base (1). These items of generic knowledge are stored in a rule base K. A user then provides the machine with some empirical assertions, contained in an empirical fact base I(t). I(t) denotes the set of all empirical observations made (and supplied to the machine) through time t. Finally, the user submits a query, meaning a question to be answered, to the machine.

Typically, a query is a conclusion whose truth is in doubt. The task of the machine is to try to deduce or refute the conclusion by applying rules in K to known facts in I(t). In a passive expert system, the result of this effort will be either a "proof" [meaning a sequence of applications of rules from K leading from the given facts in I(t) to the desired conclusion] that the conjecture in the query is correct; or a proof that it is incorrect, e.g., because it contradicts an assertion in I(t); or a statement that the machine was unable to either prove or disprove the conclusion. (In many expert systems and in Prolog, the latter two responses are not distinguished; theoretical tradeoffs between soundness and completeness and between tractability and representational power impose some fundamental limitations on the performance of any system of the type described here.) In an interactive expert system, the machine may also generate relevant questions asking the user for empirical information that will help it to answer the user's query. (A query is directed from the user to the machine; a question is directed from the machine back to the user.) If the user answers a question, the answer is formulated as a new empirical assertion and added to I(t).

As previously stated, our concern is with questions that are expensive to answer, e.g., because they require resource-consuming tests to be made. When the user can not answer questions cheaply (and may not be able to answer some questions at all, corresponding to prohibitive costs of answering), the questioning strategy that the machine uses in attempting to establish or refute a conclusion becomes critical in determining the average cost of applying it. Simple systematic patterns, e.g., based on resolution-refutation or other backtracking strategies, can be prohibitively expensive. The problem of how to adaptively generate a sequence of questions that minimizes total expected loss, assuming that an expected-loss-minimizing terminal action is taken when questioning ends, must be solved in order to evaluate the overall practical usefulness of the classification knowledge base.

To make this concrete, we adopt the following simple model of an interactive expert classification system:

- Empirical facts consist of assertions of the form T(aij) or -(aij), meaning that attribute aij is or is not present, respectively, in the object being classified. These assertions reflect (usually costly) empirical observations. (Because the prior probabilities pj

are known and stationary, there is no loss in assuming that the fact base I(t) consisting of all such empirical assertions is cleared between successive objects.)

- o Rules are of the form shown in (1). Each rule maps a conjunction of attribute values into a conclusion that the object with that attribute value pattern belongs to a specified class.

- o A conclusion is a probability distribution over possible classes, giving the posterior probability that the object being classified belongs to each class.

The expert system also has some statistical knowledge in the form of a prior probability vector p = (p1,...,pn) for the presence of each of the n attributes independently. It is assumed that p is an accurate "well-calibrated") probability model for the data-generating process.

In this model, there is only one kind of query: the user always wants to know what class an object belongs to. A question from the machine to the user is the name of an attribute whose presence or absence is to be determined and reported to the machine. The cost of testing whether attribute j is present is cj. The performance of the expert classification system is measured by the average cost per object classified -- the sum of observation and misclassification (i.e., action error) costs. It is assumed that the expert system's conclusion (posterior probability distribution for class membership) is mapped into an action in A that minimizes conditional expected loss with respect to it. We now design a questioning strategy to minimize the expected combined loss from reaching and applying a conclusion.

3. Classification Rules and Computational Complexity of MEGCP

A minimum expected-cost generalized classification problem (MEGCP) can be specified by a quadruple [K,p,c,L], where K is a logical rule base of the form (1); p is the n-vector of probabilities that each attribute j has the value "present"; c is the n-vector of attribute inspection-costs; and L = L(a,x) is the loss function. Let N be the set of n attributes that objects can have. Let N* denote the set of partial sequences of attribute values that can be generated by sequentially inspecting some of the values of attributes in N. Any sequence s* in N* has the form (a1,a2,...,aj), for some $j < n + 1$, where each ai in s* is a value (either 0 = absent or 1 = present) of an attribute in N, and where no attribute is represented more than once in s* (since its value can be inspected at most once.) A strategy or sequential decision rule for solving a GCP is a function d from N* to N u A, the union of N and A. For any partial sequence s* of length s in N*, d(s*) is either an attribute in N whose value has not yet been recorded in s*; or an action in A. If d(s*) is an attribute in N, then the strategy requires that the value of that attribute be inspected. This produces a new partial sequence of length s + 1, to which the decision function d again applies. If d(s*) is an action a in A, then d prescribes that inspection of attribute values stop and that terminal action a be taken.

Any decision rule d can be represented as a rooted tree, called a

classification tree. (We use "classification tree" instead of the more common "decision tree" to avoid confusion with the decision trees ofdecision analysis.) Each internal node of the tree represents an attribute to be inspected. Assuming that there is more than one node, the root node represents the first attribute to be inspected. Two arcs emanate from each internal node: one corresponding to an observed attribute value of 1 and the other to an observed attribute value of 0. Each arc leads to another node, representing the next observation or decison to be taken if the corresponding attribute value is observed. Each terminal node represents a terminal action in A to be taken if the sequence of nodes leading to it has been traversed. There is a one-to-one correspondence between classification trees and sequential classification decision rules d: N* --> N u A.

The total number of decision rules, or classification trees, is enormous: it is non-polynomial in n [see (11) for combinatorial and asymptotic formulas giving the number of trees generated by a given number of possible questions.] Each tree determines a corresponding expected loss. When a new object is presented to a classification tree, it can be thought of as being "dropped" down through the tree, starting from the top (root) node and following some sequence of arcs, according to the observed values of its attributes (1). Since the prior probabilities of different attribute values are known, the probability that a new object will follow any particular path (uniquely identified by its terminal node) through the tree can readily be computed as the product of the attribute value probabilities along that path. The cost of a path is the sum of the attribute inspection costs along it, plus the conditional expected terminal loss at its terminal node (found, for example, by summing L(a,x)p(x;s) over all states x that are compatible with the set of observed attribute values in the sequence s leading to the terminal node. Here, p(x;s) is the conditional probability of state or attribute combination x given the observed attribute values in s.) The expected cost for the classification tree as a whole is the sum of the expected costs for each path to a terminal node, weighted by the corresponding path probabilities. Given a specified classification tree, its expected cost can easily be computed. The problem of identifying a minimum expected cost strategy (tree), however, is provably hard except in special cases.

More precisely, given n binary random variables, the jth one having probability pj of being bj and probability (1 - pj) of being 0, where the bj are arbitrary known positive constants; and given that it costs cj to inspect the value of the jth variable, what inspection strategy will minimize the expected cost of determining with certainty whether the sum of these random variables is greater than some threshold k? It is straightforward to show that this problem, while trivial if k is less than any of the bj's (examine the variables in decreasing order of the ratio pj/cj) or if k is greater than the sum of any (n - 1) of the bj's, but less than the sum of all n of them [examine the variables in order of decreasing (1 - pj)/cj] is not in general any easier than the notorious "knapsack problem." (Take all pj's very close to 1 and consider the recognition form of the knapsack problem.) Therefore, like the knapsack problem, it is NP-complete. Moreover, this problem is a special case of MEGCP with two classes

(corresponding to the sum of the random variables being less than k and the sum being greater than or equal to k, respectively), except that all the subsets having at least k attributes present in them have not been explicitly listed in a rule base K. The implication is that the combined task of formulating and solving the corresponding MEGCP is NP-complete.

4. Algorithms for GLP and MEGLP

This section presents three classes of algorithms for the GLP [K,p,c,L]. The first calculates a conclusion (i.e., a set of class probabilities) from the statistical knowledge p and the logical classification rules in K, assuming that K is in the canonical form (1). The second class generates classification trees when observation costs can be ignored: this is the class of "recursive partitioning" algorithms that has come to dominate much of the literature on machine-learning and computational statistics approaches to classification. Finally, we present a new, computationally efficient algorithm for the case of costly observations.

4.1 Drawing Conclusions: Calculation of Class Probabilities

Given a set of logical classification rules in the canonical form (1) and a prior probability vector p for the n descriptive attributes, what is the probability of each of the classes on the left-hand side of (1)? We call this the conclusion problem: it is specified by (K,p). Note that after some empirical observations have been made on a new object, the classification rules in K can be updated and simplified to show the logical dependence of class membership on only the remaining (as yet uninspected) variables. Any rules that have become irrelevant, e.g., because the empirically observed attribute values for the object fail to match the attribute value patterns on their right-hand sides, can be eliminated from further consideration. The problem that remains, of calculating posterior class probabilities, is the same as the original one, but with a reduced set of variables. The same algorithm solves both.

The most important facts about the conclusion problem are that (i) It can only be solved at all if K is logically "complete," in a sense specified below. (ii) Even if K is complete, there is in general no polynomial-time algorithm for solving it exactly. (iii) For any pair (K,p), where K is complete, the solution to the conclusion problem can be computed by an algorithm that gives monotonically improving approximations to the correct answer. Although the exact answer can not be computed in polynomial time, close approximations can be computed with much less work.

As an example of an "incomplete" knowledge base, consider the two rules (i) C1 if [a & -b]; and (ii) C2 if [b & -a], where the class labels C1 and C2 are distinct. Assume that attributes a and b each have probability 0.5 of being present in a randomly selected object to be classified. Here, the notation is that "a" means "attribute a is present" while "-a" means "attribute a is absent." Then what is the probability of C1? There is not enough information in this rule base to answer the question. Unless we know what happens (i.e., which

class an object belongs to) if [a & b] or [-a & -b] occurs, and are willing to make the closed-world assumption that no other rules are relevant to C1 (so that, for example, there is not another rule "C1 if e,") there is no way to compute the probability of C1. In such cases, it is still possible to draw useful inferences about which classes are possible: numerical probabilities, however, require more information than is available.

A rule base K is called complete, for the purposes of probabilistic classification, if and only if it determines a unique class corresponding to any n-vector of attribute values. For example, the two rules {(i) C1 if a; (ii) C2 if -a} form a complete rule base in this sense, even though they only mention one attribute.

Completeness entails a form of consistency. An example of an inconsistent rule base is {(i) C1 if [a,b]; (ii) C2 if [b,c]}, where C1 and C2 are distinct (and where literals in square brackets are henceforth understood to be conjoined.) Suppose, for example, that a, b, and c each have prior probability 0.5. Then mutual independence implies that [a,b,c] occurs with probability 0.125. But according to this rule base, [a,b,c] implies both C1 and C2 -- an impossibility. Completeness rules out such inconsistencies. In the remainder of this section, we will assume that K is complete, i.e., that it assigns exactly one class label to each of the 2^n possible vectors of attribute values, and that the "closed-world" assumption is made that only these 2^n possible vectors are relevant.

Given a pair (K,p), where K is complete, what is the probability that an object belongs to class Cj? This question can be answered by the following inclusion-exclusion algorithm from probability theory and the reliability theory of complex engineering systems.

1. Identify all rules having class Cj on the left-hand side. Suppose there are M such clauses.

2. Form the set of all the clauses on the right-hand sides of these rules. For example, suppose this set is {[a,b],[b,c],[a,c]}, so that an object belongs to class Cj if and only if at least two out of the three attributes in {a,b,c} are present.

3. Construct the unions of these clauses taken i at a time, for i = 1 to M. For example, the unions of the three clauses in our example taken one at a time are just the clauses themselves, [a,b], [b,c], and [a,c]. The unions of these clauses taken two at a time are [a,b] u [b,c] = [a,b,c]; [a,b] u [a,c] = [a,b,c]; and [b,c] u [a,c] = [a,b,c]. The union of these clauses taken three at a time is [a,b] u [b,c] u [a,c] = [a,b,c]. Form the inclusion-exclusion series S1 - S2 + S3 - S4 + ... $\pm$ Sn, where Si is the sum, over all unions of the clauses taken i at a time, of the products of the probabilities of the attribute values in these unions. In our example, this series would be AB + BC + AC - ABC - ABC - ABC + ABC = AB + BC + AC - 2 ABC, where A denotes the probability of a (i.e., the probability that attribute a is present); B is the probability of b; and C is the probability of c.

4. The sum of this series is the probability of class Cj.

This algorithm follows from the inclusion-exclusion formula for unions of events in elementary probability theory. Its correctness can be proved by induction on the number of clauses, M.

Although the inclusion-exclusion series in principle requires the summation of 2^M - 1 terms, where M is the number of clauses in K, in fact the error inroduced by truncating the series after all ith-order terms have been accounted for (i.e., after Si) is no greater than Si, and is monotonically decreasing in i. Very close approximations to the correct probability can therefore be obtained by including only the lower-order terms of the inclusion-exclusion series, thereby saving most of the computational effort that is theoretically required for an exact answer. In essence, the bulk of the computational effort that would be required for an exact answer is concentrated where it can safely be ignored. Thus, computation of a conclusion from (K,p) can be performed effectively.

4.2 Recursive Partitioning Algorithms for Growing Classification Trees

Given a pair (K,p) with K complete, the inclusion-exclusion algorithm applied m times (once for each possible class) will compute the corresponding posterior m-vector, call it q, of class probabilities. The expected loss from each action in A with respect to q can easily be calculated, and the "optimal" action with respect to (K,p), or, equivalently, with respect to q, can be found. Let L(K,p) denote the minimized expected loss resulting from this optimal action in the absence of any additional information.

Now suppose that some empirical observations are introduced. The initial statistical knowledge base (K,p) must be updated by eliminating from further consideration all rules whose right-hand sides do not match the observed attribute values, and by replacing the probabilities pj for observed attributes with ones or zeros. Let [K,p,I(t)] denote the updated statistical knowledge base, where I(t) denotes the set of empirical observations made up through time t; and let L[K,p,I(t)] denote the corresponding minimized expected loss when the "optimal" action (with respect to [K,p,I(t)]) is taken. One possible decision rule for deciding which attribute to observe next, or which terminal action to take, given the current situation [K,p,I(t)], is as follows:

1. For each attribute that has not yet been inspected [i.e., whose value is not recorded in I(t)], compute the <u>reduction in expected cost</u> from inspecting that attribute. This reduction is given for attribute j by

   L[K,p,I(t)]-(pj)L[K,p,I(t)+{aj = 1}] - (1-pj)L[K,p,{I(t)+{aj = 0}],

   where I(t)+{aj = 1}, for example, denotes the information set consisting of all the observations in information set I(t) plus the additional observation that the value of aj is 1. Call this reduction in expected loss Vj for attribute j.

2. If Vj > cj for at least one attribute, then inspect next the attribute for which Vj - cj is greatest. This will produce a new information set I(t + 1).

3. If Vj < cj for all remaining uninspected attributes, then stop inspection and choose that act in A that is optimal with respect to the current information set, i.e., that minimizes expected terminal loss, L(a,x).

This is an example of a recursive partitioning algorithm. It generates a classification tree by successive (stepwise) myopic optimization of some criterion -- in this case, reduction in expected loss.

Different recursive partitioning algorithms can be obtained by varying the node selection criterion (here, reduction in total expected cost) in steps 1 and 2. For example, the attribute selection score Vj could be defined as expected reduction in classification entropy of the set of objects being classified at a node in the classification tree -- the criterion used for objects characterized by discrete attributes in ID3 (6); or as expected increase in the "separation" of subsets that are separated by a node, i.e., that have different values for the attribute corresponding to that node, as accomplished for continuous as well as for discrete attributes in (1-5); or as decrease in total probability (or, more generally, in expected terminal cost) of misclassification, which is one of the criteria considered in (1). Moreover, alternative stopping rules can be used to end inspection in Step 3. One possibility is to stop inspection when the best of the remaining attributes is not expected to produce a "statistically significant" difference (as defined by any of various statistical tests) between the distribution of class probabilities if that attribute has a value of 0 and the distribution of class probabilities if it has a value of 1 (6,7.) When the probability vector p is not known, but must be estimated from data, it may be desirable to reject the idea of a stopping rule altogether, and instead to first grow a classification tree of depth n (which would call for all attributes to be inspected) and then "prune" this tree back to a subtree of a size that is expected to minimize the true (as opposed to the estimated) error rate; see (1) and (5). In general, any algorithm that uses a myopic node-selection criterion as in the above examples to determine which attribute to inspect next may be referred to as a recursive partitioning algorithm; there are many such algorithms, however, and they vary considerably in their node selection criteria and implementation details.

From a decision-analytic perspective, all of these strategies can be viewed as heuristics for obtaining computationally tractable approximations to the "optimal" decision rule/classification tree. By the criterion of minimizing total expected cost, including inspection costs, all of these strategies are suboptimal -- not only because they ignore observation costs, but also because even if observation costs were all zero, the myopic optimization called for in recursive partitioning does not in general lead to globally optimal choices. Recursive partitioning continually has the expert system ask "If I could only choose one more attribute to inspect, which one would be

the most valuable one to look at?" (where "most valuable" is defined by the node selection criterion.) It then prescribes that the attribute that answers this question be inspected, unless some stopping criterion is met. But choosing the "best" attribute assuming that only one can be selected does not, in general, produce the same selection as would choosing the "best" attribute, bearing in mind that other attributes may subsequently be inspected. Nonetheless, recursive partitioning algorithms are currently the dominant way of generating classification trees in practical applications. At least two systems -- the EXPERT EASE system that implements ID3 and the CART statistical software package -- have proved that recursive partitioning is a commercially viable computational technology for many applications where large training sets and negligible inspection costs can be assumed, at least as a first approximation.

4.3 Growing Classification Trees when Information is Expensive: The Double-Sort Algorithm

We now return to the MEGCP [K,p,c,L], where K is the rule base

C1 if [a11,...,a1n]
C2 if [a21,...,a2n]
:
Cm if [am1,...,amn]

(and the literals aij within clauses are conjoined.) In light of the above comments, the following new heuristic algorithm is proposed to adaptively generate a sequence of questions for the user to answer:

Algorithm A: The Double-Sort Algorithm

1. Order the literals (the aij) in each right-hand side clause as follows: if aij asserts that attribute j is present, then define rj = (1 - pj)/cj. If aij asserts that attribute j is not present, then let rj = pj/cj. If aij asserts that the value of j (present or absent) is irrelevant, then delete aij from the clause. Sort the remaining literals from left to right within the clause in order of decreasing rj. (This is the "first sort" referred to in the name of this algorithm.)

2. Compute the expected inspection cost of each right-hand side clause, assuming that the attributes in it are inspected in the order determined by step 1 until the clause is completed (satisfied) or an empirically observed attribute value fails to match the value (present or absent) prescribed by the corresponding aij. If [a1,...,an] is an ordered sequence of attributes to be inspected, and if cj is the cost of inspecting the value of aj and wj is the probability that the value of aj will be the value prescribed by the corresponding aij (i.e., wj = pj or 1 - pj, depending on whether aij calls for aj to be present or absent, respectively) then the expected inspection cost for [a1,...,an] can be put in the form c1 + w1c2 + w1w2c3 +...+ w1w2...w(n-1)cn. In this form it can be computed extremely quickly by a single pass through the chain of successive (cj,wj) values. Let Ei denote the expected inspection cost for the clause on the right-hand side of

rule i, and let Pi denote the product of the match probabilities, Pi = w1w2...wn. In other words, Pi is the probability that the clause on the right-hand side of rule i is empirically satisfied.

3. Rank the rules from top to bottom in order of decreasing Pi/Ei ratio. (This is the second sort of the double-sort algorithm.)

4. Identify the first-ranked (left-most) attribute in the right-hand side clause of the top-ranked rule. Call this attribute j*. (Thus, j* will appear in the upper left-hand corner of the array of attributes after the two sorting operations of the double-sort algorithm have been completed.)

5. Let Vj* be the expected reduction in cost from inspecting the value of j*, as given by the formula in Section 4.2. If Vj* < cj, or if all attribute values have been inspected, then stop inspection and choose that action in A that minimizes conditional expected loss. Otherwise, ask for the value of attribute j*.

6. Delete from further consideration all rules with right-hand side clauses containing literals aij* that do not match the revealed empirical value (present or absent) of attribute j*. Update the Pi/Ei ratio for all right-hand side clauses containing literals aij* that do match the empirically observed value of j*. To update Pi/Ei, replace wj* by 1 in the formulas for Pi and Ei and delete any sunk costs from the expression for Ei. (Thus, in effect, let the new Pi = Pi/wj* and set cj* = 0 wherever it appears.)

7. Return to step 3.

As Algorithm A is executed, it produces a sequence of questions (in Step 5) that ask the user for the values of different attributes. When enough questions have been answered so that the expected costs of answering further questioning exceed the expected benefits of the additional information that will be gained from answering them, then algorithm A stops and recommends an action that minimizes expected loss with respect to what is known.

Algorithm A is designed for computational efficiency and general applicability. Its performance, as measured by the average losses it achieves compared to the corresponding theoretical minimum losses, depends on both the logical structure of the rule base K and on the cost vector c. If sufficient special structure is present, then algorithms superior to A can be developed. However, the following theoretical result suggests that algorithm A yields "good" results under some interesting conditions:

Theorem: Suppose that an object belongs to class 1 if at least k out of its n attributes have the value "present," and that it belongs to class 0 otherwise. Then the decision rule constructed by algorithm A determines which class the object belongs to at minimum expected cost.

Proof [Qiu '88 (17)]: The result can be established by showing that the sequential classification decision rule constructed by Algorithm A satisfies the necessary and sufficient conditions in (12).

As a corollary, if the conditions for proving that an object belongs to different classes are disjoint, then again the sequence of questions constructed by Algorithm A will be optimal. However, in general, Algorithm A is only a heuristic: it is possible to construct small examples in which the sequence of questions generated by the double-sort procedure is not the average-cost minimizing sequence.

The intuitive motivation for the double-sort heuristic is that the expert system should try to satisfy the easiest-to-satisfy clauses first, where "easiest" is measured by the success-probability per unit cost ratio, Pi/Ei. As soon as a clause is satisfied, a classification conclusion can be drawn, and the algorithms can stop. Within a clause, however, the hardest-to-satisfy literal, as measured by the failure-probability per unit cost ratio rj, should be attempted first. Since all literals within a clause must be satisfied to satisfy the clause, there is no point in wasting resources on the easy ones at first: if it will not be possible to satisfy the whole clause, then this fact should be discovered as soon as possible. Therefore, the most difficult parts of the clause should be tried first.

These two heuristic guidelines lead to a computationally efficient procedure (Algorithm A) for identifying the next attribute to ask about. The double-sort procedure is much less myopic than recursive partitioning algorithms, since the selection of the next attribute to ask about (the one in the upper left-hand corner of the doubly sorted array of aij's) has been heuristically optimized within and between clauses. However, the heuristic buys computational ease at the expense of global optimality in general. It ignores informational spill-overs among clauses in assigning priority scores -- e.g., situations in which even failure part way through one clause may be very valuable because of information that it yields that reduces the expected costs of resolving the remaining clauses. Ignoring these informational interdependencies makes the double-sort algorithm tractable both for generating entire classification trees a priori (an "open-loop" inspection and classification strategy) and for generating questions adaptively as the values of an object's attributes are sequentially discovered (a "closed-loop" or on-line inspection and classification strategy.) But it leaves room for improved special-purpose algorithms.

5. An Example

To understand how algorithm A works, consider the following problem. There are three attributes, a, b, and c, with prior probabilities of 0.1, 0.4, and 0.6 of being present, respectively. The cost of inspecting any attribute (to determine whether it is present) is 1. There are two classes. Objects are classified according to the following logical rule:

> An object belongs to class C1 if and only if at least two attributes are present; otherwise, it belongs to class C2.

What is an optimal classification tree for this problem? Assume that the cost of a misclassification error is much greater than 1, so that it is always optimal to definitively determine an object's class.

Algorithm A proceeds as follows. First, the logical rule must be translated into a canonical-form rule base, as follows:

1. C1 if (a,b,?)	4. C2 if (a',b',?)
2. C1 if (a,?,c)	5. C2 if (a',?,c')
3. C1 if (?,b,c)	6. C2 if (?,b',c').

Here, a means that attribute a is "present" (value = 1), a' means that attribute a is "absent" (value = 0), and ? signifies the don't-care condition as usual. Note that enumeration of rules like this is typically not a polynomial-time task. Once the rule base K has been created, however, algorithm A can be applied to it in polynomial time.

The first step is to sort attributes within clauses in order of decreasing (1 - pj)/cj ratio. This ratio is 0.9 for a, 0.6 for b, and 0.4 for c; and hence 0.1, 0.4, and 0.6 for a', b', and c', respectively. Thus the C1 clauses are already correctly sorted, and the C2 clauses should have their literals sorted in reverse order. The second step is to compute Pi/Ei for each of the six rules. For the clause (a,b,?), the expected cost is E1 = 1 + (0.1)(1) = 1.1, while the overall success probability is (0.1)(0.4) = 0.04. The ratio is P1/E1 = 0.04/1.1 = 0.036. The corresponding ratios for the other rules are P2/E2 = 0.06/1.1 = 0.054, P3/E3 = 0.24/1.4 = 0.17, P4/E4 = 0.54/1.6 = 0.34, P5/E5 = 0.36/1.4 = 0.26, and P6/E6 = 0.24/1.4 = 0.17. Rule 4, namely "C2 if (a',b',?)" ranks highest, and b' is the first-ranked attribute (recalling that the literals in C2 clauses are sorted in the reverse of their original order by the first sort.) Thus, the first question generated should be whether b is absent (or, equivalently, whether b is present.) (In a closed-loop implementation, calculation would stop at this point, with the first question being submitted to the user before any additional calculations are done.) If the answer is that b is absent, then the next question will be whether a is present. If a is not present, then the classification tree terminates with a determination that the object belongs to class C2; otherwise, it will ask whether c is also present, and will then assign the object to class C1 if and only if the answer is "yes." If the answer to the first question is that b is present, then rules 4 and 6 can be eliminated from further consideration, and rules 1 and 3 must have their scores updated to 0.1/1 = 0.1 and 0.6/1 = 0.6, respectively. Rule 3 is now the highest-scoring rule, so the next question will be "Is c present?" The object will be classified as of class C1 if the answer is "yes," and as class C2 otherwise. This completes the derivation of the optimal classification tree via Algorithm A. That it is in fact optimal follows from the general optimality theorem for k-out-of-n systems.

ACKNOWLEDGEMENT

I gratefully acknowledge many stimulating discussions with my colleagues Yuping Qiu and Scott Parrish of U S WEST Advanced Technologies. Proofs of the results on NP completeness and k-out-of-n optimality cited in this paper were provided by Yuping Qiu.

REFERENCES

1. L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

2. K.A. Grajski et al, "Classification of EEG spatial patterns with a tree structured methodology: CART," IEEE Transactions on Biomedical Engineering, Vol. EME-33, 12, 1986, 1076-1086.

3. L. Gordon and R.A. Olshen, "Tree-structured survival analysis," Cancer Treatment Reports, 69, 10, 1065-1069, 1985.

4. M.R. Segal, "Recusive Partitioning Using Ranks," Tech. Report No. 15, Department of Statistics, Stanford University, August, 1985.

5. A. Ciampi et al, "Recursive partitioning algorithms: A versatile method for exploratory data analysis in biostatistics," in I.B. MacNeill and G. Umphrey, eds, Biostatistics, Reidel, Boston, MA, 1987.

6. J.R. Quinlan, "The effect of noise on concept learning," Chapter 6 in R.S. Michalski et al, eds., Machine Learning: An Artificial Intelligence Approach, Morgan Kaufman, Los Altos, CA, 1986.

7. J.R. Quinlan, "Generating production rules from decision trees, IJCAI '87, Vol. 1, Morgan Kaufman, Los Altos, CA, 1987.

8. J.C. Schlimmer and D. Fisher, "A case study of incremental concept induction," AAAI-86 Proceedings, Vol. 1, Morgan Kaufman, Los Altos, CA, 1986, 496-501.

9. J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison Wesley, Reading, MA, 1984.

10. D.H. Fisher, "Knowledge acquisition via incremental conceptual clustering," Machine Learning, 2, 2, 1987, 139-172.

11. G.T. Duncan, "Optimal diagnostic questionnaires," Operations Research, 23, 1, 1975, 22-32.

12. Y. Ben-Dov, "Optimal testing procedures for special structures of coherent systems," Management Science, 27, 12, 1981, 1410-1420.

13. J. Halpern, "Fault-testing of a k-out-of-n system," Operations Research, 22, 1974, 1267-1271.

14. D. Angluin and P. Laird, "Learning from noisy examples," Machine Learning, 2, 343-370, 1988.

15. N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," ibid, 235-318.

16. Valiant, L.G., "A theory of the learnable," Communications of th ACM, 27, 1134-1142.

17. Qiu, Y., "Minimum Cost Classification Problem," Research Note YQ 1-88, U S WEST Advanced Technologies, 1988.

Title: LAPS: An Assistant for Debriefing Experts
Topic: Automation Aids for Knowledge Acquisition and for Expert-system Validation and Verification

Author: Joseph S. di Piazza, Ph.D.
AI Group Systems Technology Electric Boat Division
General Dynamics Corp. Groton, CT, USA 06340

ABSTRACT

The development, testing and maintenance time devoted to expert systems (ESs) needs to be reduced radically. To this end it is advisable to build general-purpose software that interweaves the extracting, structuring, testing, and encoding of knowledge while debriefing the expert in any field. LAPS (Logic Aids for Problem Solving), such a program for assisting in the interviewing, has been successfully used to produce an ES in the domain of submarine diving officers. This automatically encoded ES was produced via each entering his responses to the LAPS series of strategic-induction, AI-jargon-free queries.

Currently, the LAPS prototype has four core sessions or functions. 1) The first, the sample solution session, is one that aids the expert to map out, in his own words, the steps that he takes to solve self-chosen problems, along with the reasons and any scope-reducing assumptions that he used. 2) The second session, called dechunking, tests the expert, guiding him to find and then fill in missing heuristic steps or model-based connections occurring in his solution path. Often these are skipped by him in an effort to solve the problem quickly. 3) The next session administers five more completeness tests of the expert's input or answers to queries; it aids him to produce in a new table all alternatives to each step--conclusion-reasons combination, or implicit frame-laden rule -- in any sample solution that he has already completed.

The expert's input or solution step -- the implicit rule -- is checked, or tested, for two more matters: its having a sufficient number of necessary, or non-replaceable, conditions. This testing is done via AI-jargon-free, or everyday-language, questions, like "Is this [conclusion-reasons combination] always true?" If not, LAPS, in its remedial mode, recommends the right fix. In this case the cure or advice is to add a reason that will make the implied rule approximate as closely as possible being always true. Then, for the fourth test, LAPS assists the expert to complete other combinations or conjunctions of values for reasons or condition attributes, in an effort to have a complete rule-set. For the final two completeness tests, LAPS aids the interviewee to produce a complete set of valid values, along with well scoped attribute names. 4) After any session the interviewee can use the fourth function of LAPS : automatically produced frame-laden rules (in M1, for now). These rules can of course be used to carry out consultation sessions, though fewer of these are needed. LAPS has itself been translated from M1 to C -- to have a very fast program on an almost universally available platform, the micro-computer. LAPS is being enhanced continually.

LAPS: An Assistant for Debriefing Experts

PART 1: INTERVIEWER PROGRAM CAN SAVE EXPERT SYSTEM DEVELOPMENT & TEST TIME.

The Next Hurdle. Expert systems (ESs) are a computer solution to the problem of the limited availability and the perishability of human expertise. Many sites, including EBdiv, the submarine-building division of GD, have ESs that are in daily use. However, in their wake there arises another problem or hurdle: they take too long to build, debug, and maintain -- much longer than necessary.[1] A grounding hypothesis of this paper is that much of this inefficiency or excessive time-debt is rooted in an unnecessary isolation or serialization of the extracting, structuring, and testing of expert knowledge (di Piazza, 1986, 1987).

Overview. In order to overcome this isolation, what was encoded during 1987 was an interviewing program, called LAPS (Logic Aids for Problem Solving).[2] This program, in its own unique way, can perform and interweave various knowledge-transfer operations: the extracting, structuring, testing, and encoding of expertise. (See figure 1.)

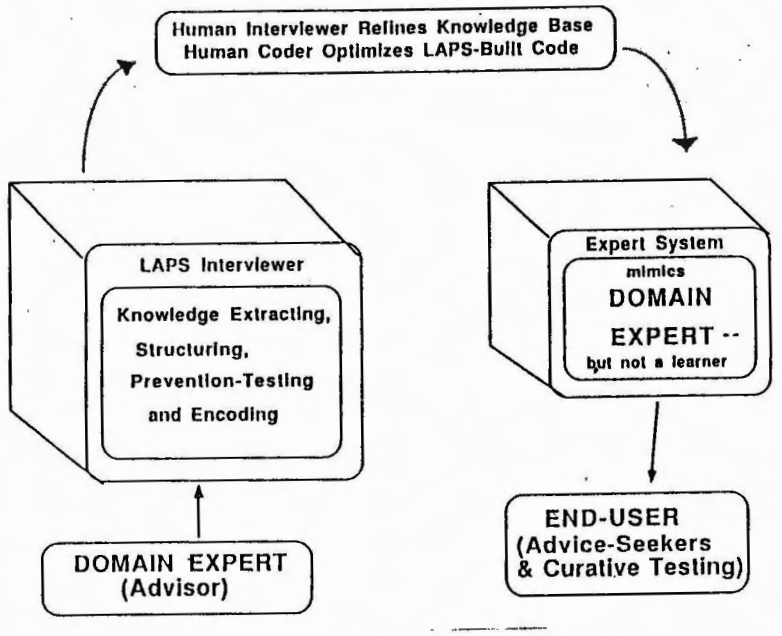


Figure 1. ES Development Cycle, with LAPS -- Reducing the Human Developer's Work

With LAPS, structuring and testing, as a way to lessen later maintainence activity, are not left as an after-thought; or else, a complete re-coding, as in the case of R1, is likely to be necessary. And via expert diving officers of a submarine responding to LAPS's queries, what was produced in 1987 was a

rigorous but restricted prototype ES in their domain, called DIVE.[3] Critically, this successful product (which is illustrated by figures 2 through 5 below) confirmed the promise of LAPS the debriefing program.

The Need for Rigorous, User-friendly Interviewing Software.

A key assumption behind LAPS is that the expert knows his domain. But he has difficulty i) expressing his knowledge completely, unambiguously, and generically, as well as ii) realizing all that he knows. For an example of how LAPS interweaves various knowledge-transfer operations, suppose that an auto-mechanic is being interviewed. Further, suppose that via his answers to queries from the interviewing software, he indirectly creates the following rule as a first try:

> If the engine's spark is irregular
> then replace the points.

The expert's diagnosis that the spark irregularity is caused by bad points would not create a problem if he were in the garage, working on a car. Before he got very far, he would notice whether or not the points were only dirty and, if so, simply clean them -- assuming that he is an honest mechanic. An ES, however, based on the above rule, would give expensive advice.

What kind of queries from an interviewing program could help to catch this error -- along with others, involving five other kinds of incompleteness?

The Conventional Consultation Session: Time-consuming and Chance-laden. Often following on the heels of the ad hoc building of many special-case but seldom chaining rules, a common method of testing is sole reliance on consultation sessions. The expert plays the role of a user of the system, has in mind a particular situation, and answers questions posed by the system. The expert then checks the advice given by the system for correctness. Many, many situations -- potentially, an infinite, hence, unreachable number -- must be tested before one can be confident of the advice given by the ES. If an ES using the above rule is submitted to a consultation session, and if the expert has in mind burned out points, then the deficiency in the rule would escape detection for the moment. It is only when he happens to try the situation with dirty points that he discovers the deficiency. Thinking of this counter-case would be a matter of chance. Indeed, he may never try a particular counter-test-case! Consultation sessions, though, are still useful as final, or back-up, confirmation.

Interviewer Assistant Saves Development and Testing Time. But prior to use of this time-consuming and uncertain cure-centered tool of repeated consultation sessions, what is needed is a less time-costly, systematic prevention-centered way i) to test rules as they are created. In addition, what is needed is

ii) to produce efficiently a system of chained rule-sets and frames. Finally, along with these features, what is needed is an easy way for the expert iii) to generate (indirectly) rules in the first place as well as iv) to have them encoded automatically. Clearly, the aim of the interviewing-encoding program is to save on development and testing time. Critically, the use of LAPS cuts down severely on the number of consultation sessions needed.

Two Projected Interviews: Prevention-assisting Dialogue. Error-prevention-assisting interviewing techniques offer no guarantee of perpetual success. Still, as consciously designed for this purpose of testing at the right moment and in a familiar tongue, they offer a higher probability of success than "leaving matters to chance." A prevention-assisting dialog might go something like this:

I(nterviewer -- looking beyond the expert's leaps in logic and wanting a complete list of solution steps): Can you insert a step betweem the following two statements?

Spark is irregular.
?
Replace the points.

M(echanic): Check the points.

The above dialog removes the ambiguity indirectly. This removal is done by reproducing more closely what the expert actually does in practice in the garage, but fails to recall -- better, delineate -- when interviewed.

The overall technical approach for this project, in part, stresses transforming interviewing techniques and the results of conceptual analysis that are found to be successful off-line. These materials are transformed into an intermediate stage involving the building of simulated dialog-laden screens. A review by the experts of the simulated screens and any subsequent changes amounts to a code-less form of rapid prototyping. After the user review, the acceptable screens are then taken as the basis of an effort to produce the code controlling the dialog on the screen.

The second simulated interview session below was sketched, in the spirit of the technical approach, as a steppingstone to the coding of LAPS. This interview takes a more direct approach than that of the interview cited above:

I (testing for insufficient conditions -- another kind of incompleteness): Can you think of a situation in which the "engine's spark is irregular" but in which some other alternative to "replace the points" is advisable? (The words in quotes are just fill for place-holders.)

M (probably surprised over his learning about his own added

knowledge): Yes.

I (extracting or prompting for knowledge, after setting the stage): What?

M: Clean the points.

I (structuring or guiding the input of knowledge): Replace the X and Y in the template below:

```
            If    the engine's spark is irregular
            &               X
            then  replace the points.
      But   if    the engine's spark is irregular
            &               Y
            then  clean the points.
```

M (responding to in-ordinary-English prompts with his familiar-to-him jargon):

```
            If    the engine's spark is irregular
            &     the points are burned
            then  replace the points.
      But   if    the engine's spark is irregular
            &     the points are dirty
            then  clean the points.
```

It is important to observe that the above dialogs, in light of the placeholders like X and Y used, could be carried out in complete ignorance of the domain. This fact offers hope for LAPS being a general-purpose automated aid to the human interviewer.

The Future Roles of the KE and ES Programmer: Greater Challenges.

Not surprisingly, the knowledge engineer (KE), or human interviewer, will probably continue to be present at LAPS sessions to clarify any over-looked ambiquity in the LAPS queries. He will be present until, as we found, the expert has "gotten the hang" of LAPS. And he seems to catch on quickly. But never does the expert have to contend with the need to learn specialized AI jargon, about frames or search or control strategies; nor does he ever have to listen to the jargon of symbolic logic about sufficient conditions, the contrapositive, the proof by resolution, etc. And the KE never need speak in these strange tongues to the expert. There are other reasons to keep the KE in the loop -- better, in the higher levels of the spiral of development. For he is needed for more nuanced, conceptual-analysis interviewing (see "Future Work" below). Moreover, the ES programmer is still needed to optimize the LAPS-built rule-base for execution speed as well as possible replacement by equivalent procedural or function-laden code (see "The Fourth LAPS Function: Code Generation" below).

PART 2: PROGRESS ON LAPS: FOUR CORE SESSIONS, WITH SIX TESTS

Our current LAPS prototype program manages a core of three interviewing sessions. These are concerned with i) extracting sample solution paths from the expert, ii) aiding him to dechunk or unpack those paths, and iii) aiding him to build a complete, rigorous set of alternatives to each step in any path or a representative path. A fourth key session, not an interviewing session, iv) automatically generates rules at the end of any previous interviewing session, if requested; it also displays and runs them if also desired, as the basis for a consult session.

Session 1. Sample Solution Path.

Initial experiments with a dialog to aid the experts to do backward chaining, had negative results: the KE may do goal-decomposition easily, but experts generally do solve problems using forward reasoning. As a consequence, we determined that the initial session on knowledge extraction, pertaining to cases, should allow the expert to express in his own words his own solution path or steps to sample problems of his own choosing. A sample solution path consists of starting statements followed by the steps that he takes on route to his final conclusion. The appearance of the resulting table is reminiscent of the traditional high school geometry proof or a university logic proof. (See Figure 2.)

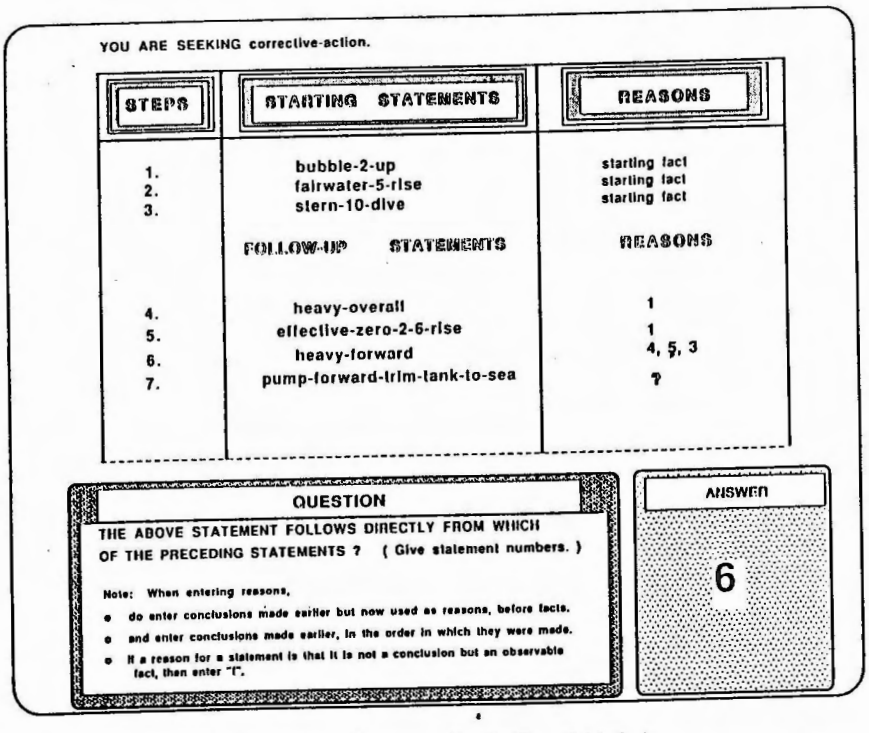


Figure 2. A Sample Solution Path, Composed by a Diving Officer.

The starting statements are facts that are true of, even define, the selected special-case problem. And they serve as "givens," driving in a forward way the reasoning process. The intermediate

steps can be conclusions, tests to be performed, or test-results. Even when the intermediate steps, such as tests to be done, are goals, they drive the reasoning of the expert in a forward way. Associated with each statement is a list of reasons in the form of numbers for statements already occurring on the list. Also extracted are i) the general category covering all solutions to those problems to be considered, ranging from "repair" to trial-and-error "solution" [= found] and ii) any simplifying assumptions used to keep the problem scope manageable. In the example shown in Figure 2, the goal category is "trimming action" or "corrective action"; and one of the simplifying (scope-reducing) assumptions is "ship is holding depth at a constant speed". The queries used to extract the sample solution path are couched in everyday language. The expert is free to run as many paths as he desires in an effort to get the better, more representative paths for further development.

It is not necessary to understand the content of the above table, only its structure. For those interested, "trimming action" pertains to a diving officer's efforts to restore the ship to neutral bouyancy. The "bubble" is a slang expression, used by submariners. It is analogous to the bubble in a carpenter's level, expressing the inclination of a submerged submarine above or below the horizontal. In the example, the submarine is inclined 2 degrees up. Water moving past the upward-inclined hull pushes up on the heavier than water ship, keeping it at a constant depth. (A lighter than water ship would require a compensating "down" bubble.) The fairwater plane is a control plane or surface, located close to the middle of the ship. Functionally, the two planes must cooperate: they must be pointing in the same direction for the force on ship to be determinable. And a third gauge on board indicates the angle of the stern plane, which is at the rear of the ship and which has an even more subtle influence on the bouyancy of the ship.

Session 2. Dechunking the Sample Solution Path.

The purpose of this subsequent interviewing session is to extract steps that the expert-user may have omitted in the original solution path. This session not only i) indirectly resolves ambiguities as in the first auto-mechanics dialog above, but can also ii) extract other "hidden knowledge", in the form of heuristics or an underlying functional or causal model. This session uses systematic and intensive questioning to achieve these purposes, far beyond that exhibited in the minor dechunking done above in the first interview of the mechanic. The thorough but clear debriefing can, for instance, focus on smaller combinations of reasons, if available. If an already entered step --e.g., ship-heavy-forward-- follows from three reasons, will any smaller combination of them lead to an intervening step? In this example, the answer is yes, since statements 7 and 3 lead to the just entered step 8. (See Figure 3.)

Comparison of Figures 2 and 3 shows the insertion of three new steps: step 4, incorporating a screening heuristic, often unmentioned; step 5, explicitating part of an underlying functional, common-sense physics model or corollary; and step 8, expressing an intermediate step, acting only like a sub-total of sorts.[4] Dechunking can be labeled the first completeness test in LAPS, for the aim for the interviewed expert is to extract a sequence of steps as complete as the ultimate end-user's needs demand.[5]

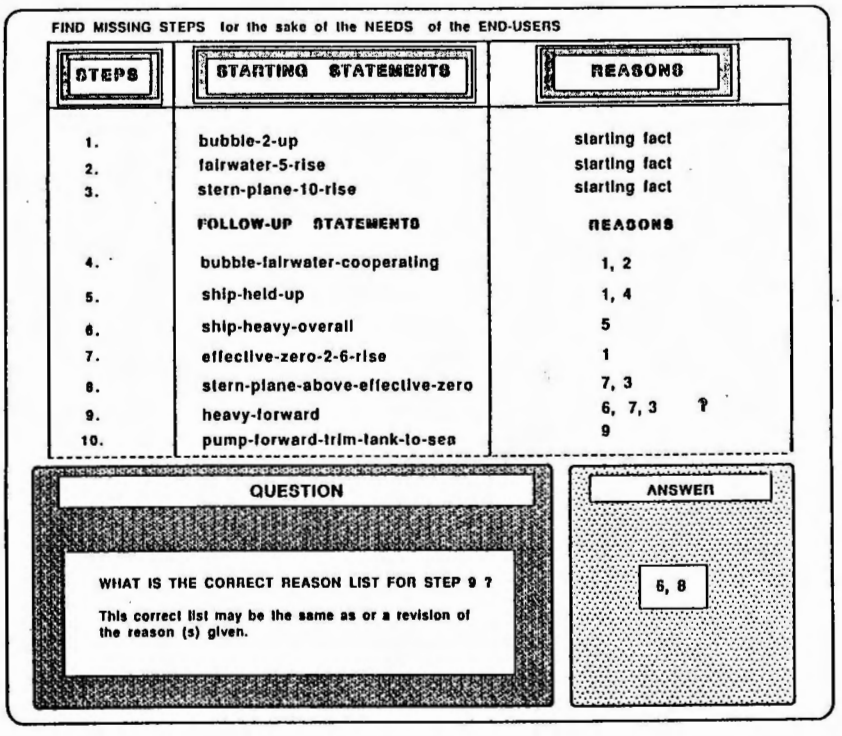


Figure 3. Dechunked Solution Path, Composed by a Diving Officer

Session 3. Alternatives Tables.

This session is the last and the most analytical of the three current sessions. It manages the development of a set of new tables, one for each conclusion-row extracted by either of the above functions or sessions. When a new alternatives table is completed, it should contain a complete set of alternatives (see the complete table in figure 5) corresponding to a particular conclusion step. In the current prototype, the table has two or three condition columns and one conclusion column.[6]

When, however, the table is first displayed, it contains a single row ("bubble-2-up" leading to "ship-held-up" -- see figure 4A). This row corresponds to a single step in one of several sample solution paths. The implicit special-case rule being built here is one that LAPS can display on the screen and that desperately needs to be generalized:

If bubble-2-up
then ship-held-up.

The First Transformation: Find the Headers or Attributes. The expert is lead by LAPS to transform the entries in the row cited above into two rows: a title row and a modified row 1. (See figure 4A; for the results of the transformation, see the same two rows in figure 4B.)

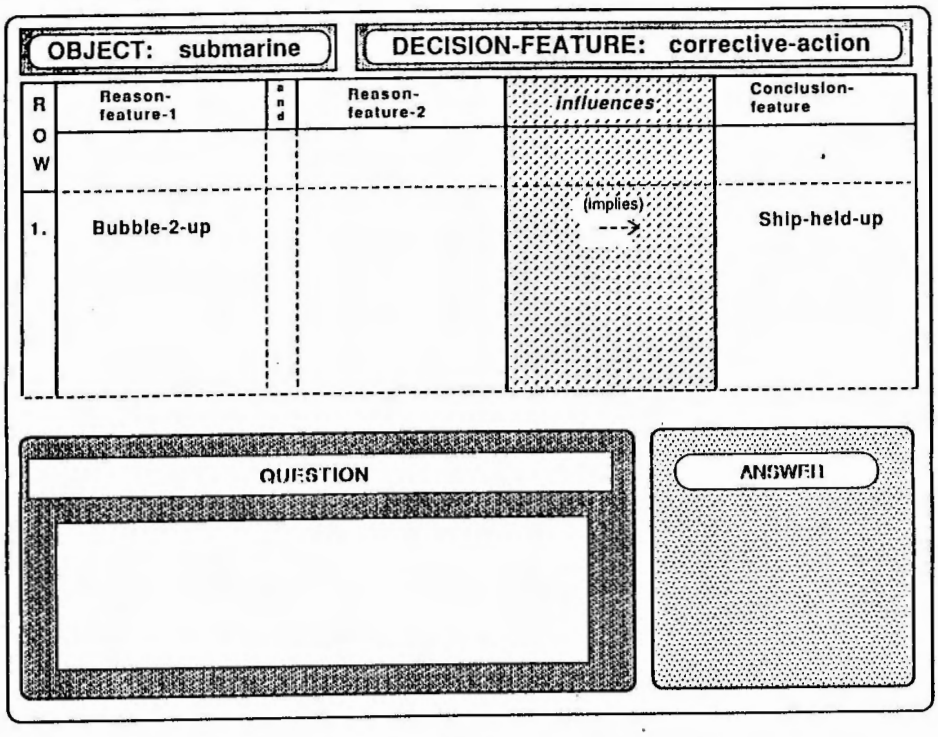


Figure 4A. Alternatives Table. Initial Table, Confronting a Diving Officer.

Types of Completeness Testing in Session 3. In this session, the expert is subjected to a complete arsenal of testing in an effort to assure a complete rule set. The first two completeness tests used in Session 3 of LAPS were developed with consideration for some concepts of formal logic (hence, Logic Aids for Problem Solving): the necessity and sufficiency of the conditions of an implication (or row). The challenge, however, is to communicate these logic requirements through ordinary language -- a challenge not met in many logic texts. To look ahead, the entire table is tested in three more ways: for completeness of valid values, completeness of combinations of valid values, and for accurate, or commensurate, scoping of attributes used in the inference-frames.

Completeness Test 2. Necessary Conditions: Beyond Special-case Rules. Special-case rules, like the one above, do properly arise in a setting like a LAPS sample solution session. But they must be surpassed; else, as with R1, they grow too numerous, weighing

down the rule base. Necessity testing (n-test) asks whether or not a condition can be replaced by another -- better, by even a set or group of conditions -- and still leave the rule true. In the example illustrated in Figure 4B, since the diving officer recognized that 2-up can be replaced by 3-up, 4-up, etc., and still the same conclusion would be yielded, this condition fails the test for necessity.

The fix suggested by LAPS to the user in this case is to "broaden the statement" (such that the value used is no longer replaceable). A diving officer then indirectly built the following M1 rule:

```
If      bubble = X-up
then    force-on-ship = up.
```

(See row 1 of Figure 4B.) This test seeks completeness insofar as a necessary condition can replace all the conceivable special-case conditions.

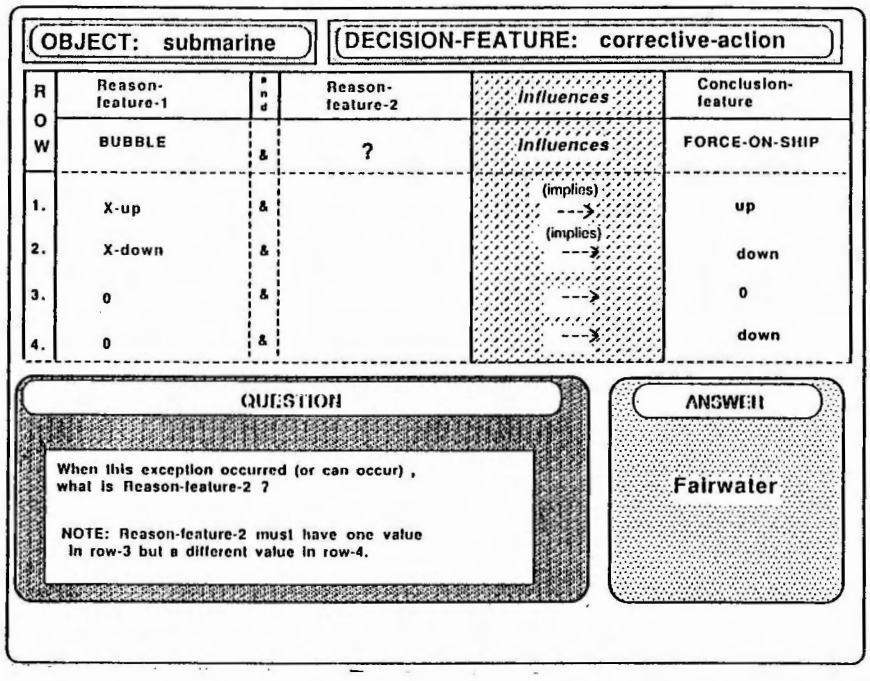


Figure 4B. Alternatives Table: After a Third S-test Question

A Series of Follow-up Queries for Clarification. The interviewee does not understand immediately all the initial LAPS AI-jargon-free queries. Then, in the case of each of the six tests, there must be a line of progressively more simple or inductive, but economic, follow-up questions, posed on-line or at least off-line. For necessity-testing,, among the graded series of the follow-up queries is "Is such and such [the converse] always true?" The converse, " if force-on-ship = up, then bubble = 2-up" is itself not always true. Hence, the follow-up query to this follow-up is again: what broadening of "bubble = 2-up" would make the statement of the converse always true?

Completeness Test 3. Sufficient Conditions. The sufficiency test (s-test) asks whether or not the user-expressed set of conditions is enough to warrant the conclusion (or to keep a person from jumping to a conclusion). An initial version of the test question is, "Can the same conditions yield a replacement conclusion?" The above rule, though having a necessary condition, fails this test; and the fix suggested by LAPS for the rule is to add another condition, or column, which would as closely as possible make the implication always true. (See figure 4B.) A follow-up everyday-language question here is "Is that [conclusion-reasons combination] always true?" One might recall the earlier, longer (pre-tabular) interview excerpt pertaining to the auto-mechanic. (There are four different generic kinds of counter-examples for n-testing and s-testing.)

Here the diving officer proceeded to add an entire column of values. (See figure 4B.) Now the transformed implicit rule for the initially troublesome row, which LAPS can display, reads:

> If bubble [of the submarine] = X-up
> and fairwater = Y-rise
> then force-on-ship = up. (See figure 5, rows 0 and 1.)

Moreover, after more than one rule has been entered, LAPS is to flag automatically rules having symptoms exhibiting lack of necessity or sufficiency: different conditions but same conclusion; same conditions but different conclusion.

Completeness Test 4. Combinations of Valid Values. As soon as more than one column appears, LAPS is designed to invoke a reasoning-aids mechanism. Then, generated are combinations of all the current valid values for the multiple condition columns -- combinations not already cited, or responded-to, by the expert. This action is taken to aid the expert in insuring that all rows (rules) have been included in the table (See figure 5).

A comment on row 5 of figure 5, wherein the bubble and fairwater angles are opposite: then, to eliminate the conflict and the indeterminacy of taking any final pumping or flooding action, the diving officers note that the bubble must be reduced to 0 -- before proceeding to find a trimming solution. This control heuristic was often not mentioned by the interviewees; but it was in some cases pried out belatedly (see row 5), if not caught earlier (see row 4 of figure 3). LAPS has been designed to have a series of follow-up logic traps to catch completeness, or material-logic, errors.

Completeness Tests 5 and 6. The Complete Set of Valid Values, along with Attribute-scoping. At some point the expert answers that there are no more rows (rules) -- reasons-conclusion combinations. Then LAPS, beginning test 5, asks the expert to confirm that there are no more valid values for each of the reason column headers. For more assurance of the completeness of

the sets of valid values, there are other follow-up tests; one is to find missing leaves in binary-tree representations of the valid values. Moreover, as a sixth kind of test, the accuracy of the column names, or attributes in ES terminology, can also be examined. In particular, the complete set of valid values can have even one member that is excluded by an attribute name; or the name can be such that at least one invalid or irrelevant value falls under its umbrella. Then the expert is urged to adjust the suspect or misleading name, respectively, for either narrowness or over-broadness.

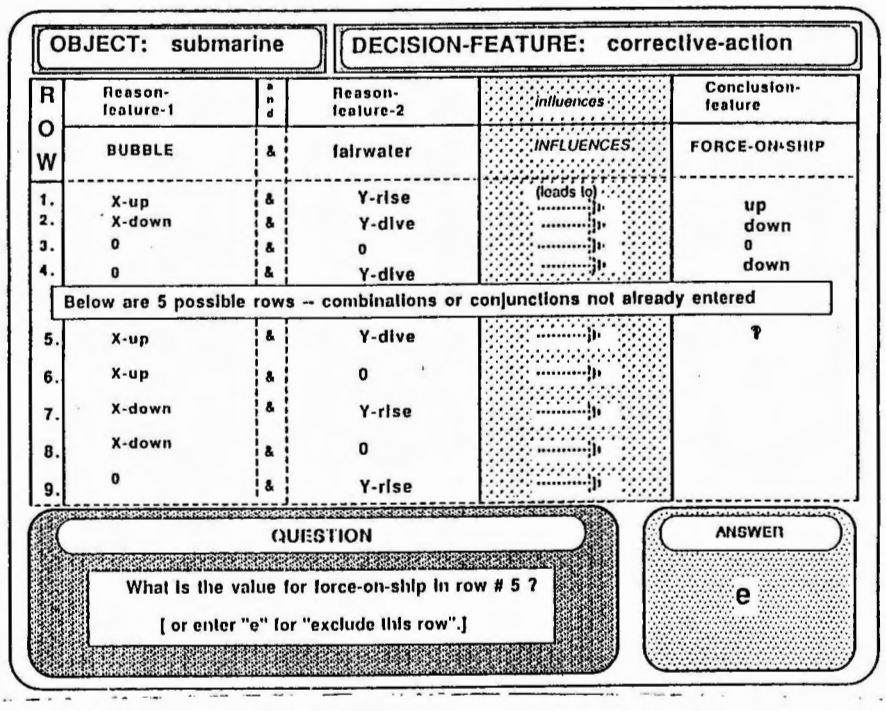


Figure 5: Alternatives Table. Additional Rows Generated.

The Fourth LAPS Function: Code Generation. The target language (or inference engine) used for running the ES rules was M1, produced by Teknowledge, Inc. Not an interviewing session or function, this fourth LAPS function is the generating of frame-laden rules from the expert's input. This function, if invoked, can be used following any of the above sessions, allowing for a consultation session at any time. But when LAPS is used, there are fewer consultation sessions needed. In other words, there are fewer times when, during a consult session involving hundreds or more rules, an inadequacy of some rule, perhaps entered months ago, is later noted. This inadequacy is signaled by the expert reporting that the ES arrived at the wrong decision in a case he has not tried previously.

Other Programming Points: The Coding of LAPS and the ES Programmer. After the ES programmer reviews the LAPS-built ES rules, he can look forward to advanced coding work: in fact, LAPS the tool is a model for re-coding application ESs. For the original M1 rules of the rapid prototype of LAPS have been translated into Microsoft C, i) with much, much faster run-time

as well as screen management but with much slower code development time [9] and ii) with very great, PC-based availability.

SUCCESSES: NOVEL FEATURES

What is novel about LAPS? To our knowledge, no other program as a whole is available for structuring, testing, and encoding an expert's input while interviewing him. The following sections of LAPS appear novel as well: the relatively analysis-free sample solution path; dechunking the sample solution, for unspoken heuristics or hidden models; and the table of alternative steps, which are analyzed via tests of five different kinds of incompleteness.

It is important to note that the testing of LAPS revealed it to be an effective error-prevention-assisting, apparently general-purpose knowledge-transfer tool; for the experts interviewed were able to respond readily to the (everyday-language) prompts. Moreover, the end-result spoke in favor of LAPS as a systematic as well as user-friendly tool. For the trim-analysis ES, built by diving officers and by LAPS, was tested for six kinds of incompleteness while it was being structured and encoded: missing steps (heuristic or model-based assumptions); special-case values; missing condition-attributes (columns); missing valid values; missing combinations of values (rows); and narrow or over-broad attribute-names. [10] (LAPS tools for overcoming the fourth and sixth kinds of incompleteness are validation tools. The others are verification tools.) Lastly, much of the inspiration for this seemingly pioneering project can be traced to insights from the dialogues of Socrates, the original expert on knowledge extracting and testing techniques via the use of a graded series of strategic-induction questions (Plato, Meno). [11]

Future Work. 1988 will be devoted to achieving enhancements in LAPS, such as

i) creating a user option to develop the alternatives tables in a depth-first or breadth-first manner;
ii) making the program easy to use by those other than the developers -- by making demonstration discs, etc.;
iii) possibly creating different modes of LAPS for different arenas of ESs;
iv) creating various tree-like graphic displays of the expert's reasoning during different sessions of LAPS, along with other user-needed graphics, menus, and windows;
v) making strategic-induction steps for extracting control rules;
vi) building a session aiding the user to select the primitive concepts -- directly or by analogical reasoning -- into which the goal-category (e.g., diagnosis, interpretation, plan-completion) of the ES can be decomposed [12];
vii) decomposing the sample solution session into even simpler-

to-respond-to sessions;
viii) building via LAPS even bigger, high-demand ESs, mainly as testbeds;
ix) incorporating into the LAPS code new kinds of sessions (in addition to the current three), more reasoning aids, and more of

the graded series of follow-up queries and logic traps, along with the fleshing-out of many stubs-to-the-user, or unworked-out high-level functions having intermediate results but no processing as yet; and
x) coding an interface to shells in addition to M1, with the accent being on low price, availability and power of the shell.

NOTES

1. Witness i) the unmaintainability of R1, DEC's great, pioneering ES for configuring computers (van den Brug, 1986), and ii) its consequent complete, template-laden re-coding (Soloway, 1987).
2. In her very useful broad survey on knowledge extracting, Hart (1986) describes backward-chaining-inspired software interviewers. They are, I add, like Boose's (Boose, 1987) and the similar, commercial software, Auto-Intelligence. But she does not at all describe any interviewer that, like LAPS, is inspired by forward chaining (from data or goals) and that interlaces knowledge extracting, structuring and testing and that uses the style of probing initial and follow-up questions made famous by Socrates.

Boose's work (1987) is the most in-depth treatment of knowledge or expertise transfer in book form, by a practicing knowledge engineer. Auto-Intelligence (1987) is the first commercial knowledge-extraction product. Very interesting. Uses the method of triadic differentiation originated by the psychologist Kelley. For it, in the pursuit of extracting from the interviewee a backward chain of rules, repeatedly asks the interviewee to distinguish between three self-chosen values.

di Piazza 's paper (1986) contains the conceptual seeds for later work. Next year's paper (1987) contains detailed design ideas, often implemented in sequences of sample screens posing interviewing questions.
3. Many former U. S. Navy officers, including diving officers, are readily available on the staff of EB (Electric Boat Division). This fact is a chief reason for choosing this domain as an initial testbed. LAPS has also been used with a machine-diagnostic expert. In this largest U.S. submarine construction site, experts in many nautical and engineering domains abound.
4. To fail to reduce the number of conditions by finding intermediate nodes is like having many-pages-long accounting records without intermittent sub-totals: confusing. LAPS is designed to have this and other strategic illogic-blocks. Because dechunking does implicitly do modularization, it will aid LAPS to build very large knowledge-bases.

The term, "dechunking", is the opposite of chunking as described by Laird et al. (1986). The writers discuss the economizing process of leaving out intermediate steps taken while solving problems. As one becomes more experienced in solving

certain types of problems, he automatically begins to chunk his solution. To make the expert's reasoning more understandable, LAPS, however, attempts to reverse this process. The LAPS team could add a new function to LAPS to do automatic chunking, in session three: have LAPS make rules from the alternatives tables where leaf values lead to final solution values, with all intermediate values skipped.

5. Indeed, it is the needs of the ultimate end-user, or the advice-seeker of the ESs built via LAPS, however narrowly pragmatic or deep and comprehensive these needs are, that guide the design of LAPS. And these same needs should guide the degree of analysis to which the expert being interviewed by LAPS should allow himself to be prompted by LAPS in any and all sessions.

6. It is necessary to restrict the conclusion columns to one per table, or rule-set, in order to "head off" even incipient moves in the direction of spaghetti logic. The new R1 (XCON) has the same restriction controlling the templates for the encoding of rules (Soloway, 1987).

7. The interviewee quickly abstracts the common feature, bubble. He does so, after LAPS, in the name of strategic induction, urges him to consider extremely different values for cell 1 of row 1 -- rather than just any series of values at all. And, in fact, he early thought of "bubble-2-up" and "bubble-8-down". "Force on ship" was more of a challenge to reach by abstraction. The interviewee, moreover, can abstract to the object --submarine-- if prompted by the right queries, acting again as smart-induction aids.

8. LAPS, then, is to flag explicit incoherencies; but it probes for the implicit versions. Moreover, eliciting confidence factors, according to the LAPS design, is appropriate only in the situation where the conditions are diagnosed to be either insufficient or unnecessary (or both) but the expert can indicate no qualitative fix.

9. Much more strategically useful than putting LAPS on an expensive LISP machine of ours -- a fading possibility but still a possibility for another translation effort for another, smaller audience. As yet, however, C development environments do not have the richness of LISP counterparts: incremental compiling and windows. The degree to which these are needed will be seen. And some C packages may include them in the future. Microsoft windows for C is a big development-environment aid, similar to the LISP-machine window-menu system.

10. High-level shells on diagnosis, etc. [see Eshelman, L., 1986, and other articles by John McDermott, formerly of Carnegie-Mellon University and now at DEC, along with S. Marcus and other students], can be fused to a general analytical-interviewing aid -- in the future. How? Perhaps, in the spirit of setting specialized knowledge-acquisition efforts onto a fruitful step within a conceptual staircase, the following suggestion can be made: the categories used in special-purpose shells can be taken as suggested categories for the dechunking efforts. The LAPS user can then choose from among the suggestions or invent one similar to some on the list or create a totally new category. In this manner, he is not locked in, unable to tailor a shell to his unique domain. (See note 12 below.)

In a similar vein, it can be said that the work of Prof. B. Chandrasekaran, of Ohio State University, on high-level special-purpose shells, can be integrated with LAPS the general-purpose interviewer. The reason is that such shells and LAPS are not opposed but are orthogonally related. For, as Prof. Chandrasekaran in 1986 commented to me personally, my work impressed him as being focused upon building a general-purpose coherence-checking program (pertaining to explicit incoherencies and implied incoherencies). Cannot his "task-specific" shells use such coherence-checking? And cannot one locate such shells within LAPS in some way, as suggested above?

In 1988, I would characterize the current third session of LAPS as a checker for five kinds of expressed or implied incoherence within a rule-set. But I today add that the first two sessions, not conceived of in 1986, gather from the expert his rather free-form expression of his knowledge of cases and missing steps in his solving of these cases. Then, the first two sessions provide the body to be massaged by the third-session analyzer; but no body, no massage.

11. The Meno is the original classic on knowledge extracting, structuring, and testing. Recall Socrates prodding an unschooled slave boy by the right series of queries to derive an answer to an opening geometry query that he could not initially answer and that is related to the Pythagorean theorem.

Moreover, in the Republic Plato, while describing the myth of the divided line, draws a distinction currently emphasized in ES circles: that between cause-based or model-based knowledge, on the one hand, and knowledge based on effective or successful belief or even on mere associations, on the other. The Dialogues are replete with other insights for AI epistemologists in general and students of knowledge transfer in particular.

Unfortunately, in Hart's fine book there is no mention of the Socratic method of teaching -- better, knowledge extracting from the student by posing the right questions.

There is another overlooked classical source on material logic, which, unlike formal logic, deals with generic tests for the truth of individual statements: Aristotle's On Interpretation. Modern logic texts treat only formal logic, dealing with rules of inference. The father of formal logic, ironically, was Aristotle (Prior Analytics), who, again, devoted a book also to material logic. (See The Basic Works of Aristotle, ed. by R. McKeon, McGraw Hill, 1941.)

To cite a contemporary gold mine, the anthropology book by Spradley (1979) is the most comprehensive book available on effective knowledge-extracting interviewing queries; but its primary focus is not the one for LAPS: unearthing problem-solving techniques, along with testing and organizing them. Its focus is the interviewee's world as a whole (Husserl's phenomenological and eidetic reductions, to speak in terms of existential epistemology); this wider focus can set the stage for the more constrained focus of LAPS.

12. This task of conceptual analysis is a knowledge-engineering job now done off-line, after or during LAPS sessions. There will always be more knowledge-transfer work for the KE to do, then to systematize and then to automate, or add to programs like LAPS.

Special-purpose shells contain within them the fruits of conceptual analysis, for some root-category, like fault, has already been decomposed into exhaustive sub-categories, like general fault and specializing feature. (Interpretation might be decomposed into recognition and classification; recognition into observation and identification; etc.) Since the pre-determined categories of any such shell may need to be tailored, there is a need to let the user-expert and the KE review lists of decomposed concepts and then to use their analogical and inventive senses to produce the best fitting categories.

13. These follow-ups, like the original queries, will be AI-jargon-free, and will lessen the need for the AI person to be continually present for any paraphrasing, especially during the earlier sessions.

REFERENCES

1. "Auto-Intelligence User's Manual", IntelligenceWare, Los Angeles, CA, 1987.

2. Boose, John, Expertise Transfer for Expert System Design, New York: Elsevier, 1987.

3. van den Brug, A., J. Bachant, and J. McDermott, "The Taming of R1," IEEE EXPERT, fall 1986, pp. 33-38.

4. di Piazza, J. S., "An Interactive Knowledge Acquisition and Verification Expert System: A Study in Logic Requirements", a paper delivered at the Workshop for High Level Tools for Knowledge-based Systems, sponsored by AAAI and DARPA, Ohio State University, October 1986.

5. di Piazza, J. S., "Interweaving Knowledge Extracting, Structuring, and Evaluating: A Concrete Design for Preventing Logic and Structure Bugs While Interviewing Experts", a paper delivered at the General Dynamics Third Annual Software Technology Conference, San Diego, CA, October 10 , 1987.

6. Eshelman, L. and J. McDermott, "MOLE: An Intelligent Knowledge Acquisition Tool," in the Proceedings of the National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, 1986.

7. Hart, Anna, Knowledge Acquisition for Expert Systems, New York: McGraw-Hill, 1986.

8. Laird, J., Rosenbloom, P. and Newell, A, Universal Subgoaling and Chunking: the Automatic Generation and Learning of Goal Hierarchies, Boston: Kluwer, 1986.

9. Plato, Meno, in the Platonic Dialogues, ed. by E. H. Rouse, New York: New American Library, 1956.

10. Soloway, E., et al., "Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-base", Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle, 1987. XCON is a later name for R1.

11. Spradley, James P., The Ethnographic Interview, New York: Holt, Rinehart and Winston, 1979.

Domain Independent Inductive Learning of Relevance*

Soumitra Dutta

Computer Science Division
University of California
Berkeley, CA 94720
U.S.A.

ABSTRACT

Informally, *relevance* refers to the fact that certain premises are more important than others for a particular conclusion. The notion of *relevance* is important for knowledge based systems, both for reasoning and for learning. Relevance also plays an important part in knowledge acquisition. In this paper we present ILROD, a domain independent computer program that performs *logic based induction* in the language of Horn clauses for learning knowledge of relevance in the presence of arbitrary, uncertainty free, logical background knowledge. The background knowledge and facts are devoid of any partial theory of the domain and contain only a description of various *attributes* and the *objects* or *entities* for which they are defined. We present heuristics for effectively pruning the search of the *generalization hierarchy* while performing *descriptive generalization* to acquire knowledge of relevance expressed in the form of *determination rules*. These heuristics include computation of *statistical uniformities* to arrive at a measure of the *goodness* of a determination rule. ILROD has been implemented in UNSW-PROLOG and has been tested on a knowledge base containing macro-economic data about the USA for the period 1955-1982.

* This work has been partially supported by NASA Grant NCC-2-275.

Domain Independent Inductive Learning of Relevance

1 Introduction

Informally, *relevance* refers to the fact that certain premises are more important than others for a particular conclusion. For example, the *nationality* of a person is more important, i.e., more *relevant* for determining his *native-language* than some other attribute, e.g., his *height*. The notion of *relevance* is important for knowledge based systems, both for reasoning and for learning [1, 15, 20, 26]. Relevance also plays an important part in knowledge acquisition [5, 26]. *Determination rules* [24-26] form one technique for specifying relevance. It is important that we devise algorithms for the *descriptive generalization* [15] of relevance to enable the autonomous acquisition of the knowledge of relevance from arbitrary background knowledge and facts. In this paper we present ILROD [10], a domain independent computer program that performs *logic based induction* in the language of Horn clauses[1] [14] for learning determination rules in the presence of arbitrary, uncertainty free, logical background knowledge. The background knowledge and facts are devoid of any partial theory of the domain and contain only a description of various *attributes* and the *objects* or *entities* for which they are defined. We present heuristics for effectively pruning the search of the *generalization hierarchy* while performing *descriptive generalization* to acquire knowledge of relevance. These heuristics include computation of *statistical uniformities* to arrive at a measure of the *goodness* of a determination rule. ILROD has been implemented in UNSW-PROLOG and has been tested on a knowledge base containing macro-economic data about the USA for the period 1955-1982.

The layout of this paper is as follows. Section 2 discusses the relevance of *relevance* and the acquisition of *relevance*. It also draws the distinction between our work and past research in the field. Section 3 reviews some of the basics of induction in the context of this work , including the language of induction chosen, the model of generality used and some of the common methods for induction. Section 4 introduces determination rules and describes how they form a useful tool for studying the inductive acquisition of relevance. Section 5 describes ILROD, specifying in detail the actual search algorithm used and the heuristics used to control the search of the generalization hierarchy. Finally section 6 describes some of the results obtained by testing ILROD on the macro-economic database and presents some avenues for further research.

2 Relevance

In this section we first describe the importance of the knowledge of relevance and then consider means for acquiring this knowledge.

2.1 The Relevance of *Relevance*

Informally, *relevance* refers to the fact that certain premises are more important, i.e., more relevant that others for a certain conclusion. For example, the nationality of a person is more *relevant*, i.e., more important for determining his native-language than some other attribute, e.g., his height. Relevance is one kind of knowledge, which while being easy to articulate, is not easily susceptible to precise definition [5, 26]. While relevance is related to the notion of *statistical dependence*, it is also strongly tied to *causality* [20]. If all events in the world have some cause, then one event can only be relevant to another if the two events share, directly or indirectly, a common cause.

The knowledge of relevance is important for knowledge based systems, both for reasoning and for learning[1, 15, 20, 24, 26]. Typically, real world knowledge is vast and most systems have to operate in the face of real constraints such as finite memory and limited execution time for inferencing. It is important in such a situation for the system to be able to *discriminate* among the different sources of knowledge at its disposal and to use those sources which are most relevant for determining the final goal of its particular reasoning or learning process. For example, a person

[1] as implemented in PROLOG

may be described by numerous attributes (e.g., his *weight, height, eye color, parental lineage, nationality,* etc.) and it is important that the system consider only the attributes (*nationality, parental lineage,* etc.) which are relevant for determining the goal(*native-language*). In complex systems, there may be considerable gains from *suitable discrimination* obtained through a *knowledge of relevance.*

Relevance is also important for knowledge acquisition as it has been observed that people/experts can often articulate rules about relevance in the absence of a strong(implicational) domain theory[5, 26]. Thus the acquisition of relevance is a natural step in the process of knowledge acquisition, often occurring prior to the acquisition of a strong predictive theory. Numerous other examples can be given in support of the importance of relevance in different areas of AI research. A strong model of causality usually forms the basis of Explanation-Based Learning and any partial theory of causality must involve relevance[8, 18, 19]. In induction, a knowledge of *relevance* is also required for *constraining irrelevance* while constructing generalizations in the context of arbitrary background knowledge[5, 15]. Relevance is also vital for making valid analogical inferences[24, 26]. For example, from the premises that both John and Bob are British and that John speaks English and has blue eyes, we can validly infer that Bob speaks English[2] , but we cannot validly infer that Bob also has blue eyes.[3]

2.2 Acquisition of the Knowledge of Relevance

Due to the importance of *relevance,* it is important that we consider the problem of acquisition of the knowledge of relevance. It has been observed that experts/people have the ability to articulate knowledge of relevance even in the absence of a strong (implicational) domain theory. Thus one of the simplest and the most effective techniques to acquire the knowledge of relevance is to allow the experts to articulate their knowledge of relevance and suitably encode this knowledge into the system after verifying its accuracy from the domain facts and rules. A more useful technique is to build into the system a capability to inductively learn these relevancies by *descriptive generalization* from the background knowledge and facts contained in it and from observations made from the environment. This would be the preferred mode of acquisition of knowledge about relevance for an *autonomous intelligent rational agent*[31].

This seems an appropriate place to draw the distinction between this work and past research in the field. Most of the past research in machine learning have tried to explain relevance using a *theory of causality* built into the *model* of the domain under consideration [1, 2, 5, 8, 11, 19]. We assume no prior knowledge of either such a theory of causality or a model of the domain. Borgida *et al.* [1] have provided some heuristics for determining relevance : in a class hierarchy of objects, "closer" classes are more relevant, values with a high statistical distribution of frequency are more relevant, etc. These heuristics are not complete and are also not very useful both due to their imprecise definition and due to the lack of a unified model of learning in which to apply them. In this paper, we present a domain independent inductive algorithm for *learning* knowledge of relevance from arbitrary, uncertainty free, logical background knowledge and facts. We shall present precise measures for determining the *goodness* of the acquired knowledge of relevance and present heuristics which use these measures to effectively control the search of the generalization hierarchy.

Various representations for relevance have been proposed in the literature. These vary from strict implications to logical *determination rules*(DRs)[24, 26] and *dependencies*[15, 20] to different heuristic formulations [7, 11, 19]. We shall use DRs to represent relevance. Section 4 describes DRs and explains the reasons for our choice.

---

[2] nationality is relevant for determining language

[3] nationality is not relevant for determining eye-color

3 Induction

In this section, we describe certain aspects of induction relevant to this research: first, the chosen language for induction, second, the model of generality used and finally, a brief review of some of the common methods for induction. Parts of the review material in this section has been adapted from Buntine [5].

3.1 Language for Induction

ILROD performs logic based induction in the language of **Horn clauses**, as implemented in PROLOG. Horn clause logic allows the description of both *disjunctive* and *conjunctive* concepts and has a well understood means for performing *inferences*, all important for developing an effective computational model for the inductive learning of relevance. Horn clause logic is a good starting point for developing such a learning program as it can ease our task of building similar programs in more extensive knowledge representations, e.g., those incorporating uncertainty or using a different subset of first order predicate logic. A significant amount of past research in learning programs has been done in Horn Clauses [3, 4, 5] or in a similar language[2, 8, 18].

3.2 A Model of Generality

A model of generality forms an essential part of any inductive learning program. Induction involves searching for descriptions consistent with the background knowledge and facts and it is conceptually useful to consider the search space of possible descriptions to be partially ordered by their respective generalities. Past research in induction has considered models of generality where the background knowledge has been in the form of facts [21, 30], type hierarchies [12], concept hierarchies [17] and Horn clauses [3, 5]. We shall use the model of *generalized subsumption* [4, 5] for background knowledge and facts expressed in the language of Horn clauses.

Informally, a first relevance description is *more general* than a second with respect to a given background knowledge, if, in any possible world consistent with the background knowledge, the first description shall hold whenever the second does. Buntine [4] has shown that with respect to a definite clause background knowledge, a predicate definition R1 is more general than another, R2, iff for every clause in R2 there exists a clause in R1 more general than it. Further, a clause is more general than another with respect to a given definite clause background knowledge if the more general clause can be converted to the other by repeatedly turning variables to constants, adding conditions or partially evaluating by applying some clause in the background knowledge to the body of the clause.

3.3 Methods for Induction

Induction can be considered as a process of searching for descriptions consistent with the background knowledge and examples[15]. Michalski [15] has distinguished between two major types of inductive learning : *concept acquisition* (learning from examples) and *descriptive generalization* (learning from observations). ILROD performs descriptive generalization for acquiring knowledge of relevance from background knowledge and facts. Researchers in AI have studied various techniques for induction in the generality hierarchy. A number of such search techniques are based on a *specific to general search*, e.g., *climbing the generalization hierarchy* [27], constructing *maximally specific generalizations* [30] and hybrid techniques [12]. A specific to general search has the effect of constructing plausible descriptions suggested by the background description. In contrast, other search strategies are based on a *general to specific search* [13, 28, 29] which has the effect of constraining descriptions from being overly general. Researchers have coupled these two complementary search strategies into hybrid search techniques, the most widely known examples of which are *Version Spaces* [16] and *Focusing* [2]. Yet another kind of induction technique is *model directed* search [9]. However, as we are considering induction in the absence of a domain model, we shall not discuss it further.

The Marvin system [27] is an example of a specific to general induction technique based on climbing the generalization hierarchy. Constructing maximally specific generalizations(MSG) [11, 17, 30] is a more powerful[4] technique for performing specific to general search. A problem with the

[4] because it allows larger jumps to be made in the *generalization hierarchy*

MSG [5] is that it can often be too specific to be of any practical use as a lower bound on inductive hypotheses and in some instances, it may not even exist[5]. Besides specific to general search has to proceed under the *justifiability* assumption[6] and there is a need to constrain *irrelevance* while constructing generalizations in the context of arbitrary background knowledge. Thus induction algorithms incorporating MSG's work well with a limited class of background knowledge using non-recursive rules or composed of relevant knowledge only.

The process of restricting a rule by negative examples during a general to specific search is called *discrimination*. A number of different search techniques have been proposed in the literature, including breadth first searches [13, 16, 28], depth first searches [2] and beam searches [15]. A problem with the general to specific search technique is the large branching factor of the *generalization hierarchy*. Shapiro [28] has shown that a breadth first search in the case of definite clauses is very expensive. Researchers have tried to restrict the general to specific search space by using restricted languages, generality models, prior information on the structure of a set of hypotheses and heuristics. A heuristic is essential for defining the *credibility of hypotheses* [22], e.g., Shapiro [28] used the weak heuristic of preferring shorter clauses.

To learn relevance from arbitrary background knowledge in a domain independent manner, ILROD performs a general to specific search in the generalization hierarchy. Section 5.1 defines appropriate heuristics used by ILROD for pruning the search space.

4 Determinations : A Technique for Specifying Relevance

In this section, we introduce *determination rules*(DRs) as defined by Davies and Russell [24, 26] and show how we can *discriminate* among DRs [10], i.e., make them more *specific* through the operations of *exceptions* and *intersections*. We also explain our reasons for choosing DRs to represent *relevance*.

Determination rules have been introduced by Davies and Russell in the context of analogical reasoning. A DR has the logical form[7]

$$\forall\ w\ x\ y\ z\ [\ P(x,w) \wedge P(y,w) \wedge Q(y,z) \rightarrow Q(x,z)\ ] \quad \text{(i)}$$

It is written as P *determines* Q or in short as $P > Q$. An example of a DR could be

nationality *determines* language [1]

What this DR states is that nationality determines the native language of a person, i.e., two persons with the same nationality have the same mother tongue. This DR captures the *relevance* of nationality for determining the native language of a person. Of course other attributes besides nationality can also determine the mother tongue of a person, e.g., the lineage of his parents and the area in which he grew up; thus we can also have other DRs of the form

parental lineage *determines* language [2]

locale of growing up *determines* language [3]

Note that DRs , 1,2 and 3, given above are the most general logical forms of the respective relevancies. Dutta [10] has demonstrated the *discrimination* among DRs by incorporating *exception conditions* and/or taking their *intersection*. As an example of the former technique, consider DR [1]. We know from our experience that a multitude of different languages are spoken in India and thus it is quite probable that two Indians do not have the same mother tongue. This is an exception condition for DR [1] and we may rephrase this DR more accurately as

nationality (with the exception of Indians) *determines* language [1']

Note that DR [1] is more general than DR [1'] in the model of *generalized subsumption* (see section 3.2). A logical representation of a DR with an exception condition is

$$\forall\ w\ x\ y\ z\ [\ P(x,w) \wedge P(y,w) \wedge Q(y,z) \wedge (w \neq e) \rightarrow Q(x,z)\ ] \quad \text{(ii)}$$

where e represents the *exception* value(Indian) of the attribute P(*Nationality*). The generality of

---

[5] if the clause is infinite

[6] i.e., sufficient background knowledge is available from which the generalization of at least some seed facts can be justified

[7] P and Q are attribute names; w and z are attribute values ; x and y are objects or entities for which attributes P and Q are defined

the logical form (i) as compared to (ii) is obvious due to the extra restricting condition in (ii).

We can also form more specific DRs by taking the *intersection* of two DRs . Consider DRs [1] and [2]. By taking their intersection, we can form DR [4] :

nationality **and** parental lineage *determines* language [4]

What DR [4] states is that two persons with the same nationality and same parental lineage have the same mother tongue. We can similarly formulate more specific DRs by adding more conjuncts to the LHS of the above DR, e.g., by combining DR [3] with DR[4] we have the following more specific DR [5] :

nationality **and** parental lineage **and** locale of growing up *determines* language [5]

It can be easily seen that DR [4] is more specific that both its parent DRs [1] and [2] , again due to the extra restricting conditions. Similarly, DR [5] is more specific than DRs [1], [2], [3] and [4]. The logical representation of a DR with two conjuncts, $P\&R > Q$, is

$$\forall\ w\ x\ y\ z\ a\ [\ P(x,w) \wedge P(y,w) \wedge R(x,a) \wedge R(y,a) \wedge Q(y,z) \rightarrow Q(x,z)\ ] \quad \text{(iii)}$$

DRs are a useful tool for studying the inductive acquisition of relevance due to their firm logical basis and their similarity to other formulations of relevance. DRs are a weaker form of strict implication rules and functional dependencies as studied extensively in the domain of database theory [24, 26]. If we have the strict implication $P \rightarrow Q$, then $P > Q$ but the reverse is not necessarily true. Many other heuristic formulations [7, 11, 19] for relevance can also be rephrased in terms of a set of DRs. The logical nature of a DR also facilitates the domain independent logical induction of relevance. Thus we feel that DRs , while not being a complete and totally adequate representation of relevance, do provide an appropriate mechanism to initiate the development of a logic based inductive program to learn relevance.

5 ILROD : Inductive Learning of Rules Of Determinations

In this section we describe ILROD, a domain independent program that performs logic based induction of relevance (expressed in the form of DR's) from arbitrary, uncertainty free, logical background knowledge. We assume that the background knowledge consists of a set of *objects* or *entities*, each described by a set of *attributes*. An *attribute* is *described* by a *predicate definition*. Examples of an attribute are a set of clauses with the same predicate symbol on the LHS of the rule, e.g., the attribute of *nationality* may be described by the following set of clauses

```
nationality(john, american).
nationality(barry, british).
nationality(X, indian) :- born(X,india).
```

Multiple values of a multi-valued attribute are represented as a list. Other than the objects and the attributes describing them, no prior knowledge about either the domain or any complex structuring of the objects, e.g., into hierarchies or classes, is assumed to be known. Due to the arbitrary background knowledge considered and the lack of an *apriori* domain model, ILROD performs a general to specific search of the generalization hierarchy. As mentioned in section 3.3, heuristics are necessary to prune the large branching factor of this general to specific search space. Below, we first define precise measures used in formulating such an heuristic and then explain the actual inductive algorithm of ILROD. The next section discusses some of the results of the application of ILROD to a macro-economic database.

5.1 Controlling the General to Specific Search

We need to define a criterion to evaluate the *usefulness* or *goodness* of a DR to help us decide when to stop searching down some branch of the generality hierarchy. Before defining such a criterion, we need to define two particular aspects of a DR which are important for defining its usefulness.

The *partial determination factor(pdf)* of a DR is a measure of the relevance or accuracy of the particular DR. It is obtained from a statistical analysis of the background knowledge used to learn the DR . Consider the DR , P *determines* Q. Let the set of distinct possible values of attributes P and Q be PV = $\{P_1,P_2,..,P_n\}$ and QV = $\{ Q_1,Q_2,..,Q_m\}$ respectively. Define[8]

[8] where |A| represents the cardinality of set A

$$X_{ij} = \left| \left\{ x : P(x,P_i) \wedge Q(x,Q_j) \right\} \right| \quad \text{and} \quad S_i = \frac{\sum_{j=1}^{j=m} X_{ij}^2}{\left(\sum_{j=1}^{j=m} X_{ij}\right)^2}$$

Then

$$pdf = \frac{\sum_{i=1}^{i=n} S_i}{n}$$

The above definition of the pdf essentially computes the average uniformity in the background knowledge base with respect to the attributes P and Q and is not a unique definition. An alternative formulation may be more appropriate sometimes, e.g., a weighted average of the S_i may be better if the knowledge base is not representative. The intuitive notion behind the above statistical calculation is this: *if attribute P is relevant for determining the value of attribute Q, then most objects having the same value for P should have the same value for Q*. Note that this captures a deeper meaning than simple statistical correlation. If P is strongly correlated with Q then usually P > Q (or Q > P) but the reverse is not necessarily true.

Examples 1 and 2 shall help to illustrate the calculation of the pdf for DR[1] of section 4, which is : nationality *determines* language.

Example 1 : Let our knowledge base consist of 20 Americans and 20 Canadians. Among the 20 Americans, 18 speak English and 2 speak Spanish. 15 Canadians speak English while the rest 5 speak French. Then

PV = {USA, Canada} and QV = {English, Spanish, French}.
Uniformity for Americans = $S_1 = (18^2 + 2^2)/20^2 = 82\ \%$
Uniformity for Canadians = $S_2 = (15^2 + 5^2)/20^2 = 62.5\ \%$
$pdf_1 = (82 + 62.5)/2 = 72.2\ \%$

Example 2: Let us now expand our knowledge base to include 20 Indians besides the citizens of the USA and Canada. Assume that for the Indians, uniformity = $S_3 = 29.5\%$. Then the new value of pdf is, $pdf_1' = 47.34\%$

Note that the pdf (or the accuracy) of DR [1] has decreased from 72.2% to 47.34% with the inclusion of Indians in the calculation of the pdf. This was caused by the low uniformity in the sample of Indians and is indicative of the fact that nationality does not determine language for Indians. To arrive at a higher pdf value for DR [1], we could specify a threshold and ignore samples for which the uniformity is less than the specified threshold. By setting the threshold arbitrarily at 50% in example 2, we can ignore the Indians in the knowledge base and arrive at a more restricted DR [1']

nationality (except for Indians) *determines* language [1']

with the same pdf value as in example 1 for DR [1] (72.2%).

However there is an important difference between DRs [1] of example 1 and DR [1'] above. Both have the same pdf value (72.2%), but while the pdf value of DR[1] was computed by considering the entire knowledge base , the computation of the pdf for DR [1'] used only two third's of the knowledge base[9]. This brings us to the second important aspect of a DR , namely its *coverage* of the knowledge base, by which we mean the proportion of the learning sample which was used for the calculation of its pdf. Thus the coverage of DR [1] in example 1 is 100% while the coverage of DR [1'] is only 66.7%. Now we are in a position to define a criterion to compute the *goodness* of a DR :

[9] it ignored Indians

The goodness of a determination is defined as the product of its pdf and its coverage of the background knowledge base.

Such a criterion is reasonable as ideally we would like to have DR's with high values for both its pdf and its coverage. But, in practice they are usually inversely related and thus we have to strike a balance between the two. The threshold level should be set carefully as both the pdf and coverage of a DR are sensitive to it. It is advisable to adopt a dynamic thresholding strategy in which the threshold is set in response to the uniformity levels in the knowledge base.

5.2 Inductive Algorithm of ILROD

A formal definition of the problem statement is :

Given : A set of objects, O, and a set of n attributes A = $\{A_1, A_2, \cdots, A_n\}$, used to describe the objects contained in O.

To find : the attributes which are relevant for determining the value of attribute A_n.

Despite the simplicity of the above problem definition, it is quite general and with suitable modifications can be used to determine relevance in more complex situations.

The inductive algorithm of ILROD performs a general to specific search of the generalization hierarchy. The search strategy used is a *best first search* with the measure of *goodness* of a DR used as a heuristic to prune the search space. The various steps of the algorithm are described below. To enhance our understanding of the algorithm, we shall work through the various steps with a small example. Let O be a set of persons and let A = { *nationality, height, parental lineage, language* }. We are interested in determining which of the other attributes of A are relevant for determining the value of the attribute *language.*

Step 1 : Form most general DRs

This first step involves the formation of the most general DRs. It is simple step in which the system simply forms the following possible DRs:

$$A_1 > A_n;\ A_2 > A_n;\ \ldots.;\ A_{n-1} > A_n$$

For our example, we form the DRs :

nationality > language; height > language; parental lineage > language;

In the absence of any apriori theory of the domain, we can do little better than postulate this set of maximally general DR's. Step 1 essentially provides the seeds necessary for the program to *move down the generalization hierarchy.*

Step 2 : Restrict DRs via exception conditions

In this step, ILROD essentially computes the value of the *goodness* of each of the maximally general DRs found in step 1. This process of computing the goodness serves a two fold purpose : first, it discriminates among DRs of step 1 by exposing the exception conditions and second, it prunes the search space by discarding DRs with low values of goodness. The process of computing the goodness of a DR can best be explained with an example. Consider the DR: nationality > language. The computation of its goodness can be visualized in the form of a tree as shown in figure 1. There is a child in the tree (of Fig. 1) for each possible value of the attribute *nationality* (we are assuming that there are three different values). Associated with each leaf node is a set of instances which is obtained by satisfying the restricting conditions given on the arcs down from the root to the leaf node. Each of these leaf nodes generates a uniformity measure(section 5.1). The pdf of the DR is given by the average of the uniformity measures of the leaf nodes with uniformities above the set threshold. The exception conditions are generated by tracing the path from the root to the leaf nodes whose uniformities are less than the specified threshold. The coverage of the DR is computed by the number of instances in the leaf nodes with uniformities higher than the threshold. The product of the pdf of the DR and its coverage gives its *goodness.*

Assume that the *goodness* of the 3 DRs found in step 1 are as shown below (in brackets):

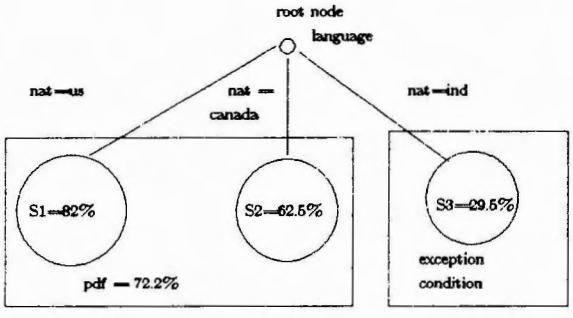


Figure 1 : Computing the *goodness* of a DR

nationality (with the exception of indians) > language (75%)
height > language (10%)
parental lineage > language (85%)

Here the assumed values of goodness for the various DRs are quite reasonable. Both nationality and parental lineage shall be relevant for determining language while height shall not be. Due to the low value of goodness for the DR: height > language (10%), we can ignore it in the future. In general, we shall need to specify a threshold value for the goodness of the DR to decide when to discard a DR. A dynamic thresholding policy that is sensitive to the goodness values of the other DRs is advisable.

Note that in this step we have already made one DR more specific (moved down the generalization hierarchy) and we have discarded one DR (pruned the search space).

Step 3 : Move further down the generalization hierarchy

In this step we perform a *best first* search of the generalization hierarchy by applying the dual *discrimination* techniques of taking intersections and generating exception conditions (while computing *goodness* values). The process of determining the goodness of a DR automatically makes the DR more specific by generating the exception conditions, if any and it also helps to prune the search space. Here we shall explain the *best first* search algorithm (see Fig. 2) of making DRs more specific by taking their intersection, i.e., by forming conjuncts out of their left hand-sides (LHS). The intuitive basis of the algorithm is that *combinations of good DRs may lead to even better DRs*. We start out with the two DRs with the highest measures of goodness and find the goodness of the new DR formed by taking the conjunct of their LHSs. If the goodness of the conjunct is higher than those of the parent DRs, then the conjunct is retained[10] and the process is repeated, else the conjunct is marked as *seen* and the process is repeated with a new pair of parent DRs. Continuing on with our example, we take the conjunct

nationality(except for Indians) & parental lineage > language [C1]

and determine its goodness. The process of determining the goodness of a conjunct can again be visualized in the form of a tree, except that now the tree has a larger depth. For the conjunct DR [C1], the tree structure has depth 2 and is shown in figure 3. If the goodness of DR [C1] is higher than either of its parent DRs, then we retain it, else we discard it. In either case, we stop our inductive algorithm as there is no additional *unseen* pairs of DRs.

At the end of step 3, ILROD produces a list of DRs with the values of their respective *goodness*. The manner in which these DRs are used depends upon the particular application utilizing this knowledge of relevance. In the absence of any domain model, steps 2 and 3 can be quite expensive for large knowledge bases. In step 2, for efficiency we can use a random sample of the large background knowledge base to approximate the actual values of *goodness* of the various DRs. In step 3, the best first search with the heuristic of first trying conjuncts with pairs of DRs with higher values of *goodness* works well in practice. For efficiency, we can terminate the search

[10] as we have a DR with a higher value of *goodness*.

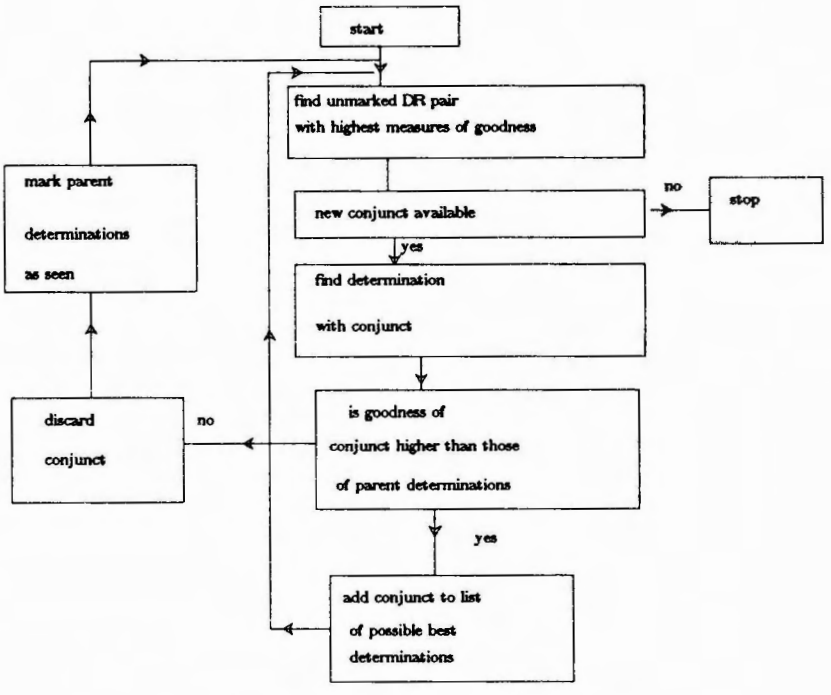


Figure 2 : Taking intersections of DRs

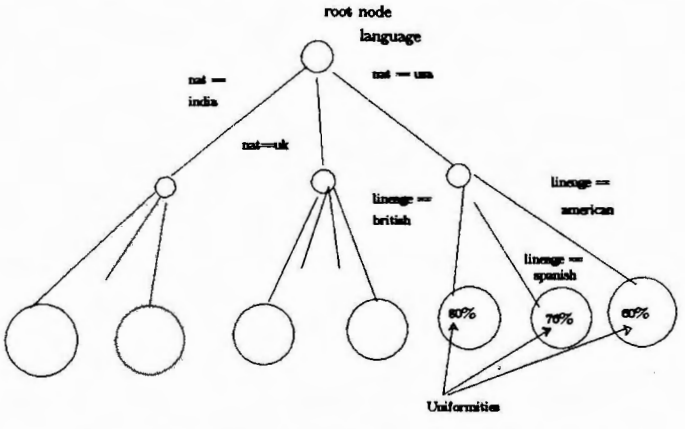


Figure 2 : Computing *goodness* of a Conjunct DR when the *goodness* of the parent DRs falls below a certain threshold.

6 Results and Conclusions

In this section, we first describe some of the results obtained by testing ILROD on a macro-economic database of the USA and then provide some brief concluding remarks.

6.1 Results

ILROD has been implemented in UNSW-PROLOG and has been tested on a macro-economic data base containing macro-economic data for the USA for 28 years through the period 1955-1982. There are 16 macro-economic variables of interest in the database, e.g., the gross national product (GNP), investment(I), consumption(C), etc. Due to a lack of space, we are unable to describe our results in detail (see [10] for details). However, the DRs found by ILROD (e.g., *change in investment* $>$ *change in consumption*) are validated by economic theory and predictions for economic variables made on the basis of these DRs fared quite well when compared with the actual values (see tables 1 and 2). Additional extensive testing of ILROD is necessary for

more conclusive results.

Table 1					
chi determines chc					
Year	Predicted values of chc			Predicted range of chc	Actual value of chc
76	53	55		(37 - 57)	56
77	53	55		(37 - 57)	50
78	37	56	52	(37 - 57)	45
79	28	27		(-7 - 29)	28
80	10	-7	21	(-7 - 29)	4
81	37	37	52	(37 - 51)	27
82	10	21	21	(-7 - 29)	14

Table 1 compares the actual values of the %change in comsumption(chc) with those predicted by the %change in the investment(chi) for the years 1976-1982. The predicted values are found by analogy with the DR *chi determines chc* in the background which was found by using data for the years 1955-1975. The values of both chi and chc have been scaled up. The predictions are good for all years except for 1981.

Table 2					
chgnp determines chpo					
Year	Predicted values of chpo			Predicted range of chpo	Actual value of chpo
76	38	36		(35 - 39)	34
77	35	37	35	(35 - 39)	34
78	38	36		(35 - 39)	33
79	35	37	36	(35 - 39)	30
80	33	34	34	(33 - 34)	26
81	35	37	36	(35 - 39)	27
82	33	34	34	(33 - 34)	25

Table 2 compares the predicted values of the %change in the potential output (chpo) by the predicted values of the %change in the gross national product(chgnp) for the years 1976-1982. The predicted values are found by analogy with the DR *chgnp determines chpo* in the background which was found by using data for the years 1955-1975. The values of both chpo and chgnp have been scaled up. The predictions are good for the years 1976- 79 for the years 1980 - 82.

6.2 Conclusions

In this paper we have described a domain independent inductive program to learn knowledge of relevance from arbitrary, uncertainty free logical background knowledge. While it is not a complete program for accurately learning relevance in all situations, we feel that it is a right step in the direction of an autonomous acquisition of the knowledge of relevance. The inductive algorithm described in this paper is simple and best suited for domains about which little prior knowledge is known. Any well defined structure in the domain which may be known *apriori*

should be exploited by a system for greater accuracy and efficiency. ILROD does not take such *apriori* knowledge into account. This is one of its limitations. Extension of the ideas in this paper for more general background knowledge (e.g., those including noise) is a subject for future research. Another interesting issue is devising better representations for relevance.

References

[1] Borgida, A., Mitchell, T. & Williamson, K. ,*Learning improved integrity constraints and schemas from exceptions in databases and knowledge bases*, In Brodie,M.L. & Mylopoulos, J. Eds., *On knowledge base management systems*, pp. 259-286, New York, Springer- Verlag, 1986

[2] Bundy, A., Silver, B. & Plummer, D.,*Model directed learning of production rules*, In Waterman,D.A., & Hayes-Roth,F.,Eds.,*Pattern directed inference systems*,New York, Academic Press, 1985.

[3] Buntine, W.L.,*Towards a practical theory of Horn Clause Induction*, In Proc. of Ninth Australian Computer Science Conference, Canberra, 1986.

[4] Buntine, W.L.,*Generalized subsumption and its applications to induction and redundancy*, in Proc. of the European Conference on Artificial Intelligence, Brighton, U.K., 1987.

[5] Buntine, W.L., *Induction of Horn Clauses: methods and the plausible generalization algorithm*, Int. J. Man Machine Studies, 26, 1987, pp. 499-519.

[6] Davies,Todd & Stuart Russell. *A Logical Approach to Reasoning by Analogy.* Technical Note 385, AI Center, SRI International; June 1986.

[7] Davis, Randall , *Interactive Transfer of Expertise: Acquisition of New Inference Rules*, 5th IJCAI , 1977.

[8] DeJong,G. & Mooney,R., *Explanation based learning: an alternative view*, Machine Learning, 1, 1986, 145-176.

[9] Dietterich,T.G. & Michalski,R.S., *Learning to predict sequences*, In Michalski,R.S., Carbonell,J. & Mitchell,T.M., Eds., *Machine learning-an artificial intelligence approach*, Vol II, Morgan Kaufmann, Los Altos,1986.

[10] Dutta, S., *Inductive Learning of Rules of Determinations*, M.S. Report, Univ. Of Calif., Berkeley, 1987.

[11] Fu, Li-Min & Bruce G. Buchanan, *Inductive Knowledge Acquisition for Rule Based Expert Systems*, Knowledge Systems Laboratory Report No. KSL-85-42, Stanford University.

[12] Kodratoff,Y. & Ganascia,J.G., *Improving the generalization step of learning*, In Michalski,R.S., Carbonell,J. & Mitchell,T.M., Eds., *Machine learning-an artificial intelligence approach*, Vol II, Morgan Kaufmann, Los Altos,1986.

[13] Langley,P., *Language acquisition through error recovery*, CIP working paper 43, CMU,1981.

[14] Lloyd,J.W.,*Foundations of Logic Programming*, Springer Verlag, 1984.

[15] Michalski,R.,*A theory and methodology of inductive learning*, Artificial Intelligence, 20, 111-161, 1983

[16] Mitchell,T.M., *Generalization as a search*, Artificial Intellgence, 18, 203-226, 1982.

[17] Mitchell,T.M., Utgoff P.E.,Nudel,B. & Banerji,R.,*Learning problem solving heuristics through practice*, In Proc. of 7th Intl. J. Conf. on Artificial Intelligence, 1981, Vancouver, Canada, 127-134.

[18] Mooney, R., *A domain independent explanation based generalizer*, in PRoc. of AAAI-86, 1986, Philadelphia.

[19] Pazzani,M., Dyer,M. & Flowers,M., *The role of prior causal theories in generalization*, In Proc of AAAI-86, 1986, Philadelphia.

[20] Pearl,J. & Paz, A.,*On the logic of representing dependencies by graphs*, In Proc. of Canadian Artificial Intellgence, Montreal, Canada, 1986.

[21] Plotkin,G.D., *A further note on inductive generalization*, In Meltzer,B. & Michie,D.,Eds., Machine Intelligence 6, pp 101-124, Elsevier North Holland, N.Y., 1971.

[22] Rendell,L., *A general framework for induction and a study of selective induction*, Machine Learning, 177-226, 1986.

[23] Russell, Stuart J. *The Compleat Guide to MRS*, Technical Report STAN-CS-85-1080, Stanford University, June 1985.

[24] Russell, Stuart J. *Analogical and Inductive Reasoning*. Ph. D. thesis, Stanford University, Dec. 1986.

[25] Russell, Stuart J. *Preliminary Steps Toward the Automation of Induction*, AAAI 86.

[26] Russell, S.J., & Davies, T.R., *A logical approach to reasoning by analogy*, In Proc. of IJCAI-87.

[27] Sammut, C.A. & Banerji,R.B., *Learning concepts by asking questions*. In Michalski,R.S., Carbonell,J. & Mitchell,T.M., Eds., *Machine learning-an artificial intelligence approach*, Vol II, Morgan Kaufmann, Los Altos,1986.

[28] Shapiro, E.Y., *Inductive inference of theories from facts*, TR 192, Dept. of Comp. Sc. , Yale Univ., 1981.

[29] Shapiro, E. Y., *Algorithmic Program Debugging*, MIT Press, 1983.

[30] Vere, S.A., *Induction of relational productions in the presence of background information*, In Proc. of the Fifth Intl. Conf. on Artificial Intelligence, 1977, Cambridge, Mass.

[31] Mitchell T.M., Jamie G. Carbonell, Ryszard S. Michalski, Eds., *Machine Learning : A guide to current research*, Kluwer Academic Publishers, Boston, 1986.

A Self Modifying Rule-Eliciter

Ladan Esfahani
Dr. F.N. Teskey

ITRI,
Brighton Polytechnic
Moulsecoomb
Brighton BN2 4GJ

Abstract

This paper describes the role of KET (Knowledge Encoding Tool) in automatic rule modification. KET was originally designed to elicit rules and other forms of knowledge in a Domain Expertise. KET's main intention is Expert/Knowledge Engineer guidance in building a Knowledge Base and for this reason its rule modifications are conditional upon its user's wish. KET's description has already been discussed in detail in previous literatures[1]. The focus in this paper is on additional functions in KET, implemented to enable rule verification in a Knowledge Base. This paper also discusses KET's future learning capabilities based on Analogical methods.

1. Introduction

The problems of Knowledge Elicitation/Acquisition have been known to Expert Systems builders for some time[2]. One of the basic aims of our AI environment in Brighton Polytechnic is to build tools that would aid with this task. Our AI environment is called IDEAS[3]. An Expert System Shell called RBFS[4](Rule Based Frame System) is the back_bone (and the first) of IDEAS tools which fulfilled this aim. Systems written in RBFS represent Frames and Production rules in the form of lists. In order to increase the grasp of Experts/Knowledge Engineers (we call them users from now on) on the information structures in RBFS (ie. Frames and their relationships), the need for some improvements was felt. VEGAN[5] is a tool that enables the formation of data and relationships (call it Information-Base) in a visual manner. KET (Knowledge Encoding Tool) was created to facilitate building the Knowledge Base in RBFS and was, therefore, a natural step forward from VEGAN.

The role and functions of RBFS[4], VEGAN[5,6] and KET[1,6] have already been discussed in previous literature. In this paper we will focus on the self organisational issues of KET and its design on Learning. As a reminder, however, we will try to describe the tools preceding KET, very briefly. This includes a brief summary of RBFS and VEGAN's capabilities followed by a description of the original KET functions. We will then proceed to describe the rule verification capabilities in KET followed by additional issues on Learning, namely analogical methods[7]. The use of laddering technique[8] has also been explored in this respect. The paper concludes with a brief discussion of advantages and disadvantages of the system.

Before describing the tools in IDEAS, it might be worth explaining our choice of representation in these tools. This was partly because of the familiarity of some of our Domain Experts with graphical symbols (eg. their use of Critical Path Analysis[9]). The problems that exist with the use of Natural Languages[10,11,12,13] also discouraged us from incorporating this feature in our tools.

2. Background

In this section we will try to give a very brief description of RBFS and VEGAN. These tools' capabilities have been put forward in a very simplistic way here. For your own understanding and in order to do justice to these tools please refer to the appropriate references.

2.1. RBFS, Rule Based Frame System[4]

RBFS is an intermediate system between an Artificial Intelligence Language and a Knowledge Engineering shell. As its name suggests, it uses a combination of Production Rules and Frame Based techniques[14] for its Knowledge Representation. RBFS was initially inspired by EMYCIN[15], hence some similarities exist between RBFS and EMYCIN's functions. One of these is the format RBFS employs in order to write Production Rules (ie. use of Premises, Predicates and Certainty Factors). Some of RBFS capabilities include the use of "facets" (one way of implementing meta knowledge)[16] for: a) propagating default values, b) inheritance of values from parent Frames to their children and c) other customary value assignments. The latter includes value derivation via questioning the client, inference via Production Rules and procedural attachment. RBFS also allows complex instantiation (ie. instantiation of a chunk of inter-related concepts) within a domain model.

RBFS has been successfully used by Knowledge Engineers in academic and commercial environments. Its application domain spans such fields as project planning[17], bid proposals for flight simulators[18] and designing functional test strategy for printed circuit boards[19].

For clarification purposes, two main distinctions has been drawn within RBFS. That is between the Information-Base (ie. data and relationships) and knowledge (eg. production rules). In order to increase the accessibility of the Information Base in RBFS, the notion of Associative Networks[20] was introduced and used. VEGAN was a tool whose objective was just this.

2.2. VEGAN, Visual Editor for Generation of Associative Networks[5,6]

VEGAN was created in order to give some visual expression to the Frames in RBFS in the form of Associative Networks. VEGAN has allowed modelling Information in RBFS in the form of statements of facts and possibilities. It allows the user to generate general and specific nodes (ie. objects or concepts) in their appropriate contexts (ie. planes). Other VEGAN capabilities include filtering of planes, modelling inheritance, the use of instantiation directives, allowing network animation and automatic placement of instantiations. In future, VEGAN's development will be directed towards the control mechanisms and the use of a critique in modeling the information domain.

When the problem related to modelling information was dealt with using VEGAN, our line of investigation was channeled towards finding a way to alleviate the problem of encoding rules and knowledge in general. As a result, a course of research went underway towards the provision of graphical tools for visualisation of the rule-base. This has led to two different approaches. KET is a tool which directly encodes the Knowledge Base via a graphical representation. Another approach[22] is to use conceptual graph[21] representations for generation of production rules.

3. KET, As a Tool for Encoding Knowledge[1,6]

KET was originally designed in order to help the Domain Experts/Knowledge Engineers in building a Knowledge Base by providing an easily used graphical tool rather than an Artificial Intelligence language.

The initial objectives for the design of KET were;

1) To simplify the creation of Production Rules.
2) To visualise the interdependence of rules and their contents.

The latter need was expressed by Knowledge Engineers who used to make a dry run of this process using paper models.

These objectives were approached in the following fashion.

To satisfy objective (1), different modules of RBFS's Knowledge Base (including the production rules), were designed as miniature "forms" in KET to be filled by the user.

To satisfy objective (2), the use of tree-like structures seemed inevitable. However, in order to maintain consistency with VEGAN (which is a prerequisite in building a Knowledge Base in our environment) the use of Associative Networks was retained. The combined representation allowed the depiction of object-relationships in the form of an Associative Network, embedding in themselves the appropriate modules (forms) dictating their value derivation method. Any ambiguities in these modules (eg. slots with no values) could be further clarified by extension into object-relationships and their description modules (forms) and so on. This mechanism enables the users to trace the relationships and the source of their reference in a Knowledge Base. A KET diagram forms a tree structure with "Ask" modules as the leaves of the tree and, "Rule","Proc" and relationships as internal nodes.

"Rule" (Fig. 1.a), "Ask"(Fig. 1.b) and "Proc"(Fig. 1.c) refer to derivation of values in slots (ie. properties or attributes) via inference of rules, asking directly from the client and calculating by procedures, respectively.

Features described so far in KET assist the Expert/Knowledge Engineer with encoding a Knowledge Base. However, at this stage of implementation, the system still relied on the user for modelling the domain and checking of the rules. Overall, this used to be a lengthy process.

4. Graphical Implementation of Learning

The authors started to investigate the existing machine learning methods with an emphasis on graphical systems in order to overcome the problems in checking rules and modelling domain expertise. We will discuss the issues in Machine learning used in KET's present and future design. Next section will describe the rule verification issues in KET. The section following that is a description of analogy and analogical methods in Learning, and how and to what extent these could be implemented in KET in the near future.

4.1. Rule Verifications

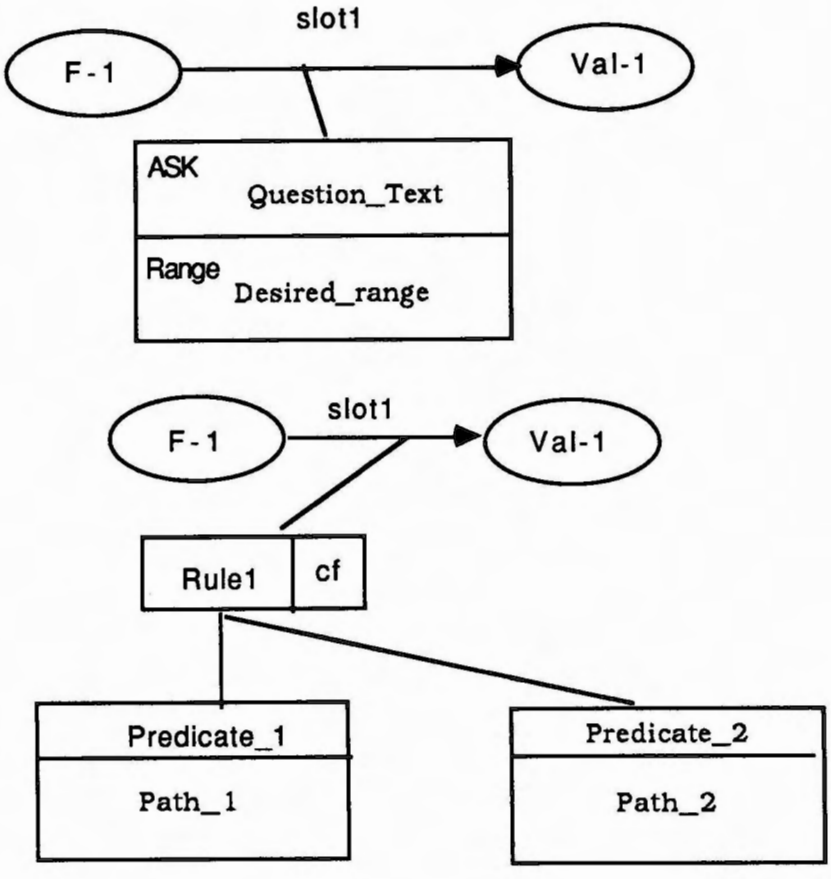
The learning features in KET/VEGAN point to the system's ability to modify its own Knowledge Base. These features include basic rule checkings, such as consistency, recursion and conflict resolution, as well as abstraction of new intermediate concepts. These points were specifically clarified by Luc Steel in his discussion of second generation Expert Systems.[23]

In the following we will describe these features in relation to Expert Systems in general, followed by their representation in KET. Please note that at each stage, the system will modify what it diagnoses as an anomaly, after reporting it to the user. Modification will not take place against user's wish.

In the following illustrations "F" stands for Frame, "Val" for Value and "CF" for certainty factor.

4.1.1. Consistency

Consistency in an Oxford dictionary is described as : "not contradictory" or "constant to same principles". In a Knowledge Base this can be interpreted as uniqueness of definitions for concepts or uniformity of the rules. Consistency as far as building a knowledge base using KET is concerned would mean searching for slots with duplicate assignment of their value derivation modules. Searching for slots which do not have a value derivation method defined for them is another kind of consistency check. So you could say that the system is making sure that all the slots in the rule-base have *consistently* been defined (ie. *all* have been defined in *only one* way). In the following example, a certain slot (generally called slot1) has two different modules for deriving its value. One is direct enquiry from the user (via an Ask section) and the second is via rules. These methods are defined in different parts of the knowledge base. In such cases the system reports the findings back to the user. Also if the premise of a rule contains a path whose member does not have a means of value extraction, that is brought to user's attention, too.



4.1.2. Recursion

Recursion is not always a bad idea in programming, for example recursion used to calculate Fibonnarchi numbers in Pascal programming. However, recursion occurred among the rules of a knowledge base means rules which use concepts whose inference will refer back to themselves. This is an undesirable state of which the user should be aware. In KET, checking for infinite self recursions, involves looking for concepts whose related rules contain antecedents that refer to the original concept. However, recursion can occur in more than one level. ie. rules that refer to concepts which refer to other rules which at some point refer to the original concept. In future, more in depth scrutiny will be catered to diagnose recursion of this nature.

4.1.3. Conflict

Conflict in Oxford dictionary has the following meaning: "opposition of incompatible wishes". Conflict among the rules in Knowledge Base terminology means rules with identical conclusions and different premises (cf. incompatible wishes!). A number of conflict resolution methods have been used in the past. Some have suggested the chronological ordering of rules of this nature, each firing in turn and depending on their success rate, gaining priorities in their specific category.

In KET's representation, the triplet (Frame-slot_value) points to the conclusion of a rule. The predicates represent the antecedents. However, since more than one rule could be attached to a single concept (ie. relationship or link), and each rule could have several predicates, checking for the antecedents of conclusions has to be divided into two parts; a) checking for the rules related to relationships and b) checking for the predicates included in the rules. However,

conflict resolution has already been catered for in RBFS and that is done by examining the certainty factors attached to rules. If a rule has a certainty factor 1.0 or -1.0, then that rule will be the first to fire as its success or otherwise could determine the outcome of the conclusion without testing the remaining rules. Therefore, conflict resolution is carried out by assigning priorities to rules according to their certainty factors.

4.1.4. Redundancy

Having located antecedents of concepts as rules and predicates, the thing to do next in redundancy check is to search for all the rules attached to *a particular* slot. Identical rules are then identified and reported to the user (see figure (a) for illustration). The second stage involves searching for *different* slots that have rules with identical premises attached to them (see figure (b) for illustration).

figure (a) : Equal rules attached to the same relationship.

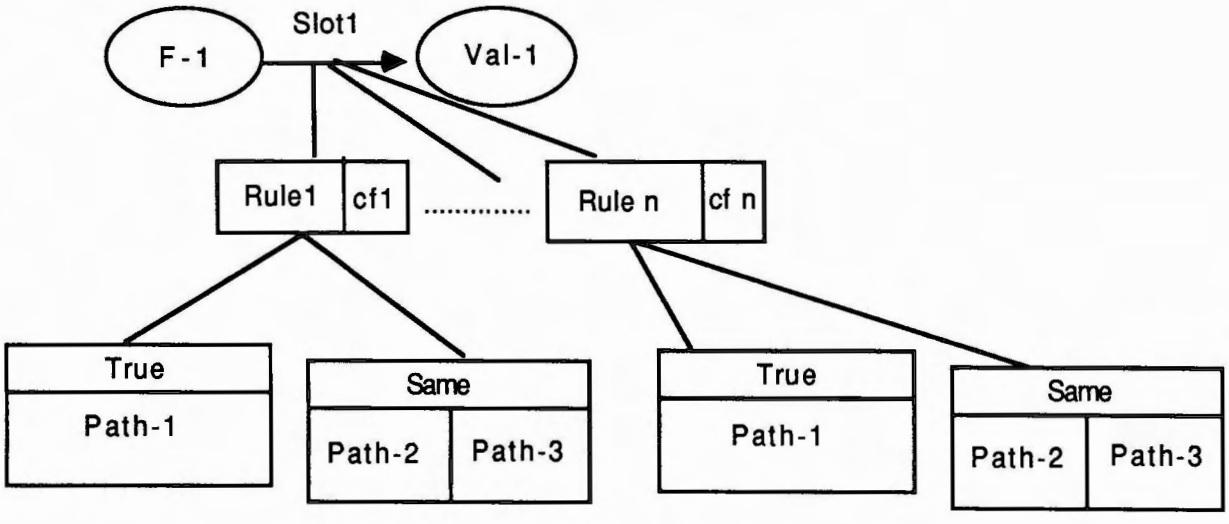
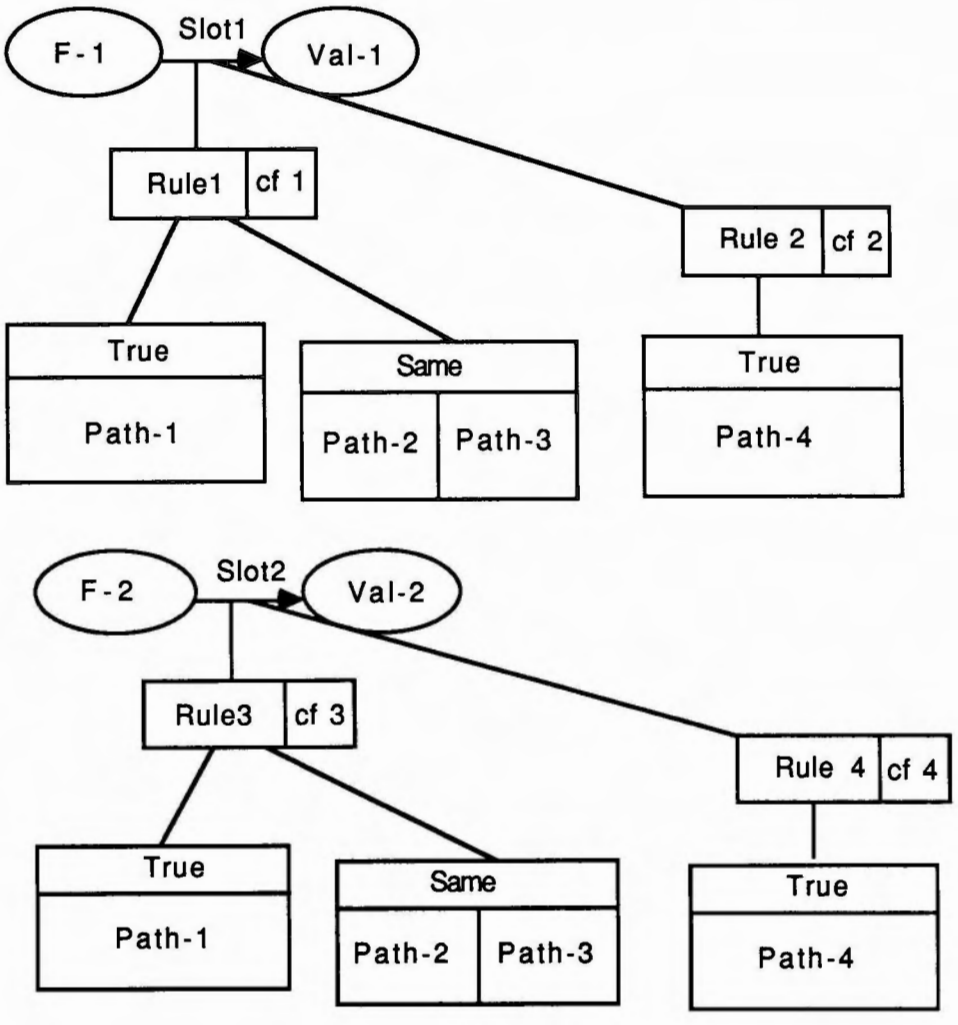


Figure (b) : Equal rules attached to different relationships.



Should the system be granted permission to modify these conflicts, it will ask the user for the least favorable concept or rule in order to eliminate. When this name is provided (represented by the name of the rule in case(a) or the relationship in case (b)), then that concept with its rules and predicates or the rule with its premises (as the case maybe) will be deleted from the network. If the redundancy is of case(b) and the slot's name is used in any other part of the knowledge base, then that name is replaced by the name of the concept with which our deleted concept was incompatible. If, however, the conflict is of type(a) then the least favourite rule is simply deleted.

4.1.5. Abstraction of Concepts

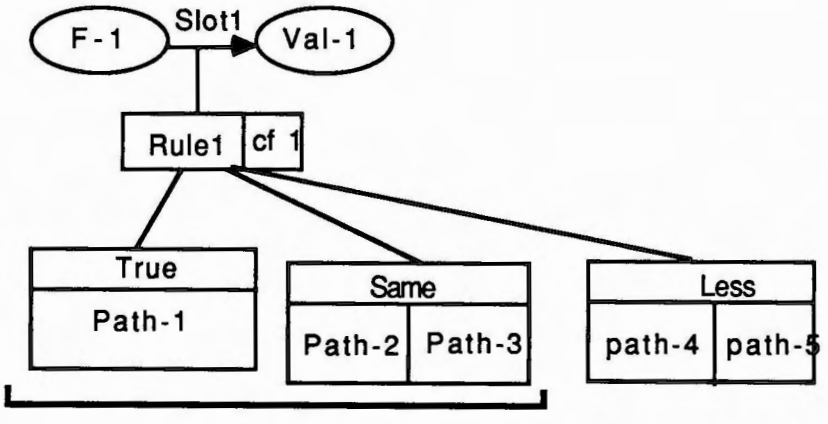
Abstraction of intermediate concepts was designed as a means of condensing and in turn tidying up the rules. This feature also helps with conceptualisation of higher concepts. In conventional rule-based systems, rule condensation could take place when two or more rules have an identical subset of conditions. In such cases a new rule could be abstracted with this subset as its premise and its conclusion would be user defined. These subsets of conditions can then be deleted from the original rules and in their place will be a new antecedent (ie. the new user defined conclusion). Consider the following example:

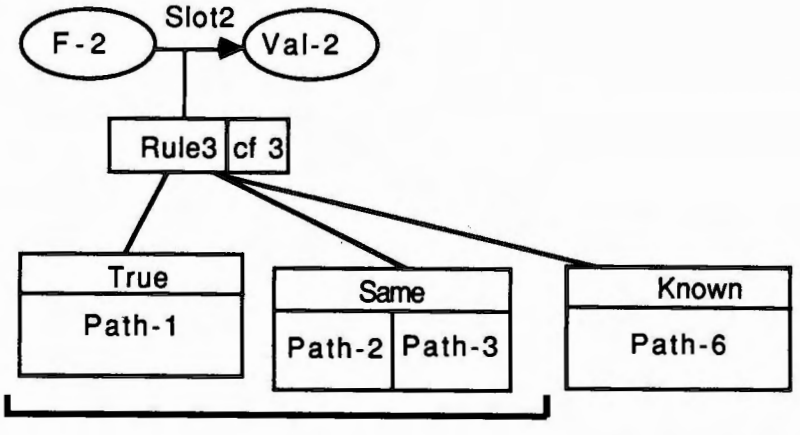
Rule 1 : If Condition_1 and Condition_2 and Condition_3 then conclusion_1.
Rule 2 : If Condition_1 and Condition_2 and Condition_4 then conclusion_2.

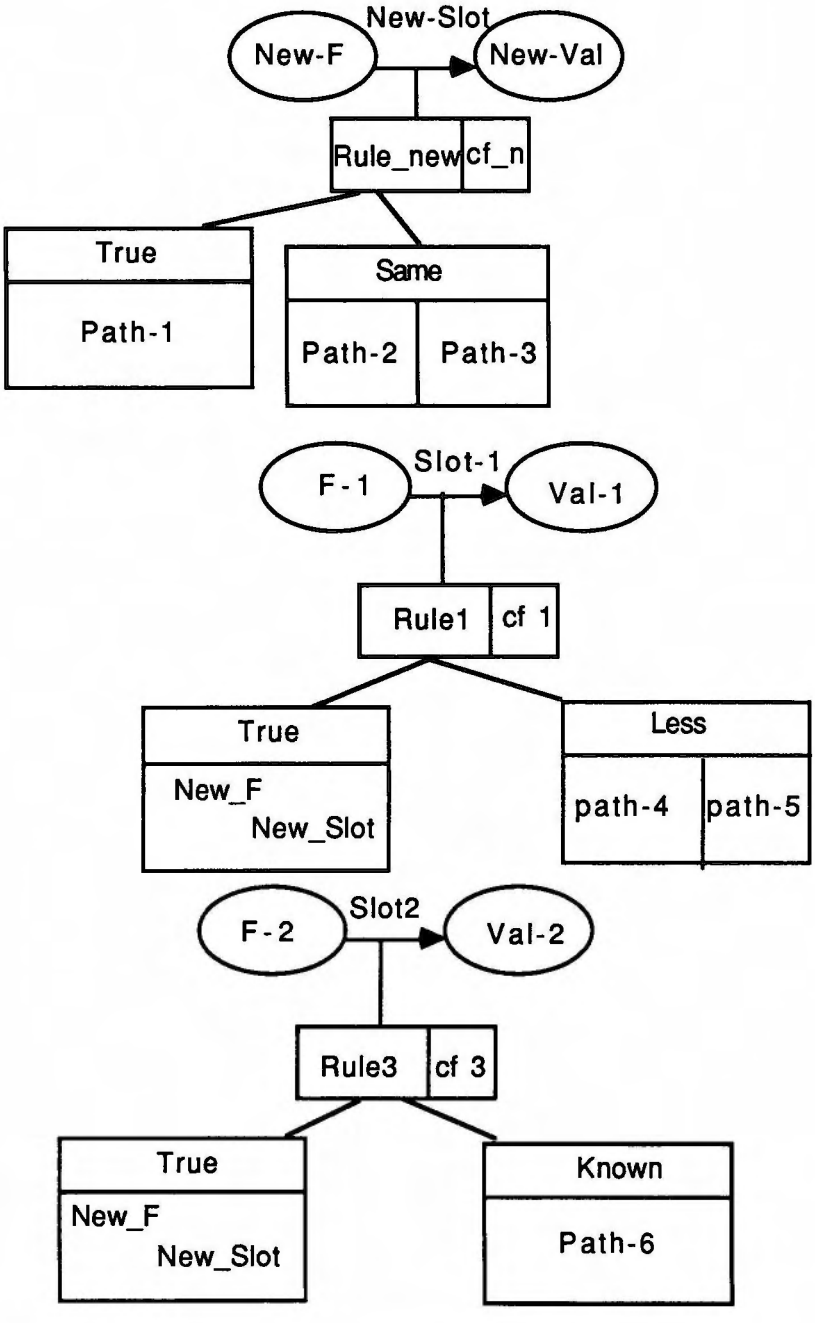
Abstraction of a new rule could be realised as follows:

Rule3 : If Condition_1 and Condition_2 then User_conclusion.
New Rule 1 : If User_conclusion and Condition_3 then conclusion_1.
New Rule 2 : If User_conclusion and Condition_4 then conclusion_2.

In a KET diagram, the conditions are the predicates used in a rule-premise, and the conclusion part is the (Frame_Slot_Value) triplet (ie. the conclusion of the rules). The following will represent a generalised form of abstraction of this type. The diagram in this page indicates a situation where the possibility of such abstraction is diagnosed. Next page's diagram illustrates the rule-base after such abstraction.







So in above, predicates "True" and "Same" were identified as identical subsets in rule1 and rule2. These predicates were consequently deleted from these rules and used as the sole premise for a new section. The new section would acquire its triplet of (Frames-slot-value) from the user (ie. New_F, New_Slot and New_Val). The substitute for "Same" and "True" predicates in the original rules, will be the predicate, "True", whose condition includes the new concepts, ie. New_F and New_Slot.

The factors mentioned in this section (4.1.) are now implemented in KET.

4.2. Analogy and Learning

KET's ability to modify rules was the first step towards a learning system. However, after these implementations the problem of domain modelling remained a major source of problem for our users. It would be helpful if the system could work out the general domain model rather than the user. In order to solve this

problem we turned to search for the most effective techniques in learning. Learning by analogy was mentioned time and again, in human cognitive research, human learning and Machine Learning literature. But what does analogy mean? What is its implications in machine learning or Expert Systems?

4.2.1. Definition of Analogy and Its Role in Learning :

The definition of analogy in an Oxford dictionary is : "agreement, similarity, (to, with, between); process of *reasoning* from parallel cases; resemblance of function between organs essentially different."

However, this definition would not offer a satisfactory description when applied to Machine Learning field. Keane[24] states a broad definition of analogy as: "An Analogy is product of certain cognitive processes in which specific coherent aspects (ie. causally-integrated aspects) of the conceptual structure of one domain are matched with and/or transferred into another domain. In the transfer of these specific conceptual aspects, relations will tend to be important, although attributes may also be important."

This definition has been derived following experiments carried out on humans in order to understand humans' analogical Learning. Understanding the analogical thinking in humans can be useful if we were to design effective analogical methods in computer systems. The evidence suggests that in humans the processes underlying analogical thinking are those which make use of relatively concrete prior experience. Keane[24] in his survey of analogical reasoning states that this concrete prior experience is a contrast with more abstract type of knowledge which underlies standard understanding. He defines the characteristics of the standard understanding as: "application of generalised knowledge to target situations which are familiar in order to (a) fill in the inferential gaps in one's representation of these domains (b) to make predictions about what may happen next." Point (a) is now designed in the form of laddering technique (see below) which is of value to KET . However, prediction of future events would be relevant to those applications which the tools in IDEAS are not suited (ie. Point (b)).

The above, describes the application of analogical reasoning in situations where the generalised knowledge is clearly structured. However, there exist situations which cannot be handled with the existing generalised knowledge. Such situations trigger a mode of non-standard understanding using analogy, according to Kean. These situations are further divided into two categories : 1) unstructured situations, ie. novel or unfamiliar situations. In this case a person uses analogy in order to *borrow* knowledge from known base domains. 2) incomplete situations which are relatively familiar and can be categorised by generalised structures, but individuals do not know enough about the domain to determine what should be done or how it may be better understood (ie. one's knowledge of domain is incomplete). In the context of Expert Systems this type of analogical learning would not apply to KET or shells intended for expert's use. Experts, in general, usually know their domain well!

Keane basically has focussed on analogical methods in problem solving. However, we also use analogy to structure concepts in our mind as well as solving problems. R. Forsyth[25] gives a good example of analogy of this type. With a particular insect in mind, we focus on its activity of feeding its infant with bait. It is the insect's habit to leave the bait outside the nest before entering the nest in order to arrange it in a desired state. Experiments show that by moving the bait away from the nest's entrance, the insect attempts to bring the bait back to its previous state, and enters the nest to rearrange it once more. When the experiment is carried out several times, the insect does not find a solution to its problem (eg. it does not learn to bring the bait inside the nest before rearranging it). This shows

that the insect is unable to recognise (retrieve and map) the similarities between the past and the present situations and consequently does not conceptualise a new state in its mind to overcome this. An example of a repetitive state in a knowledge base can be found in its rules' antecedents. Conceptualising this repetition has already been taken care of in the form of abstraction of intermediate concepts in KET (ie. abstraction of identical conditional subsets mentioned in section 4.1.5).

So far two applications of analogical learning has been identified in connection to KET. They are a) abstraction of concepts via recognising repetitive conditions or identifying the similarities of structures in a domain model followed by b) "filling the inferential gaps in one's representation of this model". Part of case (a) has been taken care of in section 4.1.5. The following describes a design to satisfy the second part of case (a) (ie. abstraction of general concepts from similiar structures) and case(b).

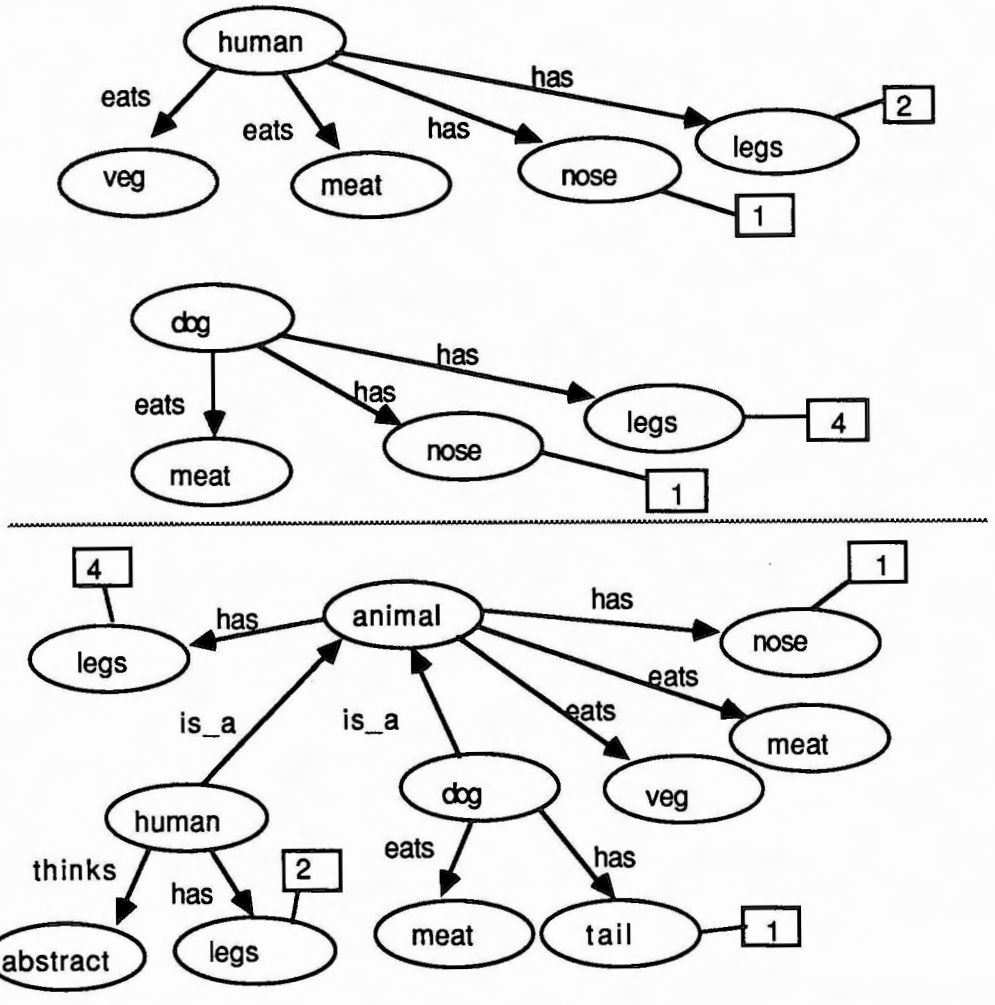
4.2.2. Learning in KET, inspired by analogical methods :

Our aim in using the learning methods in KET is to aid the users with domain modelling. In order to provide the users with some primary hunches with this modelling, the use of induction was proposed. Induction is useful to create general nodes from examples of objects or concepts. It is, therefore, system's task to detect the similarities between these examples. The similarities considered, are those between the nodes and their relationships. After reporting these similarities and obtaining permission from the user, general concepts can be created. Naming of the generic nodes is left to the user. The given examples are then checked against their general description (associated nodes in the general plane) and further specific characteristics can be added to those original examples. In fact what is termed the laddering technique[8] is carried out in order to complete the definition of all instances of a generalised node by eliciting the characteristic of those concepts. This, to some extent, satisfies the requirement in "filling gaps in one's representation" in the context of analogical method.

The following example will illustrate this proposal. Imagine the problem of presenting the system with some primary example of the concepts "human" and "dog". The following indicates that a "dog" has 4 legs one nose and eats meat. It also indicates that "human" has two legs, one nose and eats vegetable and meat. On discovering similarities of these functional relations (eg. eats and has) between the nodes "human" and "dog", system should be able to suggest that a generalised node could be created to include "human" and "dog" as its instances. Naming of this generalised node or object is left to the user. We say that in this case user chooses the label, "animal".

At this stage, the system should also investigate further properties that each of these objects possess. For example, the user, when prompted for additional discriminatory features of "dog", could suggest that this concept has a tail. In case of humans the user could mention the ability of human to "think" in "abstract" terms. These will result in new nodes and relationships and will aid with the creation of a full network. New nodes could be created by prompting the user for further instances of the object "animal" and their properties and so on.

The following is an example of this kind of modelling. The diagrams above the dotted line are the examples given to the system and that below the line is the generalisation made by the system. Generalisations follow the rules that Kodratoff[7] has specified in this respect. eg. in generalising on 2 legs and 4 legs, value 4 is chosen, ie. that which includes a broader category is generalised.



Of course, not all the properties that a concept possess are essential for its functionality. For example it is important that a car should have wheels but its colour is inconsequential for its definition. The terminology in our graphical network has catered for these problems in terms of links which model *importance* in a domain eg. links that model hierarchical relations, such as *is_a* and *inst* links. However, we understand that the fulfillment of the present design, needs inclusion of additional standard links such as part_of, consists_of, etc. to improve on this modelling. These requirements are being met in other parallel research programmes taking place in our Research Institute.[26]

The method of generalising concepts from examples (using induction) and also completing the definition of objects (using laddering technique), fit into some applications of analogy in Machine Learning. It should be noted that Induction is defined very broadly, within the framework proposed by Holland et al. (1986) as "all inferential processes that expand knowledge in the face of uncertainty"[27] . The use of analogy, especially in the context of problem solving, is thus one instantiation of this general concept of induction.[24]

5. Conclusion

The main objective of KET is to ease encoding of the Knowledge Base and increasing experts involvements in Expert Systems' development.[28] This is a highly desirable feature in our opinion, which will encourage rapid prototyping on behalf of the Domain Expert, and will increase expert's experimentation with the knowledge base. KET has achieved this objective by providing a graphical tool with a simple representation. KET has also succeeded in making apparent the rule

dependancies in an IKBS. The Learning features in KET enable the automation of rule verification and modification within a system. The user, at all times, is in control of these modifications. In order to assist the user with some initial hunches in domain modelling, Learning capabilities such as creation of intermediate concepts, and induction have been designed. These features were inspired by analogical methods in Machine Learning. Creation of intermediate concepts is a capability that KET already possesses. However, induction and the use of the laddering technique remains to be implemented in the near future.

The advantage of KET and other tools in the IDEAS system is simplicity of their representation. However, even at this level, first time users have to be acquainted with a new taxonomy, as simple as it maybe. The existence of this taxonomy is needed if our tools are to be applicable to a wide range of domains. Because of this new vocabulary, however, a domain expert needs some time to familiarise him/herself with the system. The average time taken for a person familiar with computers but unfamiliar with our systems, to create a reasonable prototype, is two weeks.

KET and VEGAN, as an integrated system, have been applied to, and tested in, the field of project planning. VEGAN and RBFS have been generally used in a number of domains (see section 2.1). The results of these applications show a tendency towards better modelling in well defined domains (eg. engineering design, project planning). Our representations also show some similarity with the notations used in engineering and design fields. This has led us to believe that these systems are particularly applicable to synthetic problems as opposed to analytical ones (eg. law).[8]

Acknowledgement

The funding for this project was provided by NAB (National Advisory Board).

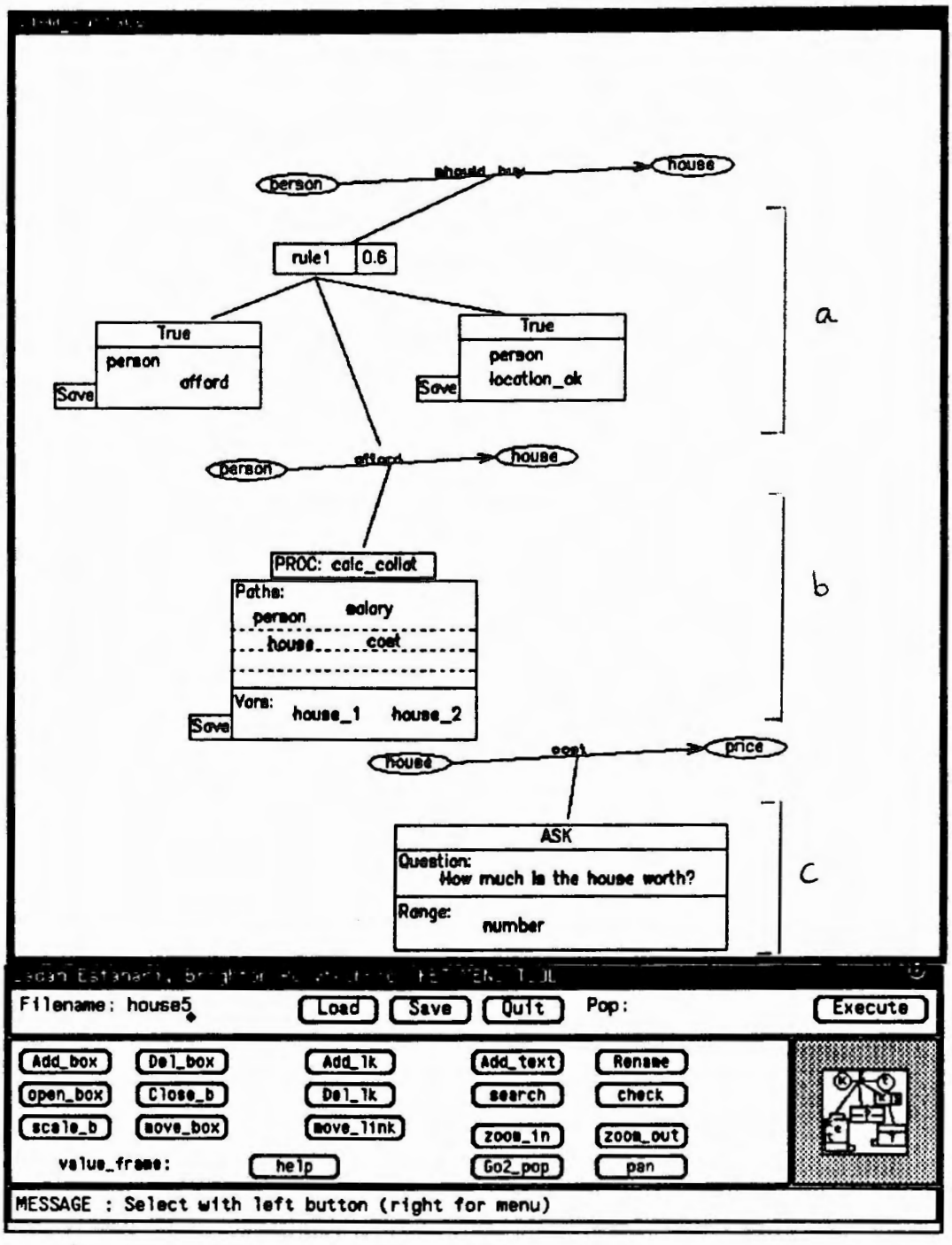


Figure 1 : A KET Diagram

References

1. Esfahani, L., Teskey, F.N., "KET, A Knowledge Encoding Tool", *Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge Based Systems*, section E5, 2-3 September 1987.

2. Hayes-Roth, F., Waterman, D.A., and eds, D.B. Lenat, *Building Expert Systems*, Addison-Wesley , 1983 .

3. Eklund, P.W., Barber, T.J., and Teskey, F.N., "An AI Environment For Knowledge Based Systems Development", *Proceedings of the KBS 87 Conference.*, pp. 127-135, On-line Publications, London, 23-25 June 1987.

4. Barber, T.J., Marshall, G., and Boardman, J.T., "A Philosophy and Architecture for a Rule Based Frame System (RBFS)", *submitted to: International Journal of AI in Engineering*, 1987.

5. Kellett, J.M., Winstanley, G., and Boardman, J.T., "A Methodology for Knowledge Engineering Using an Interactive Graphical Tool for Knowledge Modelling", *International Journal of AI in Engineering*, submitted August 1986.

6. Esfahani, L., Kellett, J.M., "VEGAN and KET: An Integrated Graphical Approach to Knowledge Acquisition and representation", *to be published in Knowledge Based Systems Journal*, September 1988.

7. Michalski, R., Carbonell, J.G., Mitchell, T., *Machine Learning : An Artificial Intelligence Approach*, Vol. II.

8. Boose, J.H., Bradshaw, J.M., "Aquinas, A Knowledge Acquisition Workbench for Building Knowledge-Based Systems", *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge Based Systems*, 2-3 September 1987.

9. Lokyer, K.G., *Critical Path Analysis : Problems and Solutions*, Pitman Publishing, 1966.

10. Shaw, M.L.G. and Gaines, B.R., "Knowledge Engineering For Expert Systems", *Proceedings of COMPINT: Computer Aided Technologies*, 1985.

11. 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 .

12. Davis, R., "Interactive Transfer Of Expertise: Acquisition Of New Inference Rules", *Artificial Intelligence*, vol. 12 , no. 2 , pp. 121-157 , August 1979 .

13. 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.

14. Minsky, M.A., "A Framework for Representing Knowledge", *MIT AI Memo*, no. 306, 1974.

15. Buchanan, B.G. and Shortliffe, E.H., *Rule-Based Expert Systems: The MYCIN experiments of the Stanford Heuristic Programming Project*,, Addison-Wesley, Reading, MA, 1984.

16. Laurent, J.P., Ayel, J., and Thome, F., "KEE, Knowledge Craft, Art.", *The Knowledge Engineering Review*, vol. 1, no. 4 , November 1986 .

17. Marshall, G., Kellett, J.M., Lim, B.S., and Boardman, J.T., "PIPPA : An Expert Project Planning System in Manufacturing Engineering.," in *Proceedings of the KBS 87 Conference*, On-line Publications, London, 23-25 June 1987.

18. Barber, T.J., "Pragmatic project control using a huristic algorithm and an intelligent knowledge-based system", *proceedings of IEE* (1987)

19. Lea, S.M., Brown, N., and Katz, T., "Knowledge Representation Automaton for Functional Test Strategies - KRAFTS", ITRI/R/12, Brighton Polytechnic, 1987.

20. Findler, N., (ed.), *Associative Networks - Representation and Use of Knowledge By Computers*, Academic Press, New York, 1979.

21. Sowa, J.F., Conceptual Structures - Information Processing in Mind and Machine, Addison-Wesley, Reading, Massachusetts, 1984.

22. Eklund, P.W., Kuczora, P., "A Conceptual Graph Implementation for Knowledge Based Systems Development", *submitted to the conference on Human and Organisational Issues of Expert Systems*, May 1988.

23. Steels, L., "Second Generation Expert Systems", *Research and Developments in Expert Systems (III)*, 1986.

24. Keane, M.T., *Analogical Problem Solving*, Ellis Horwood Series in Cognitive Science, 1988.

25. Forsyth, R., *Machine Learning and Its Applications to Information Retrieval and Knowledge Based Systems.*

26. Kuczora, P, "A Reasoning Strategy for Nonmonotonic Inheritance Systems", submitted to ECAI 88, Jan 88.

27. Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P., "Induction: Processes in Inference, Learning and Discovery", *Cambridge, MASS: MIT Press*, 1986.

28. Shaw, M.L.G., Gaines, B.R., "Interactive Knowledge Elicitation", *Canadian Information Society Session*, May 1984.

SECOND GENERATION KNOWLEDGE ACQUISITION SYSTEMS

Brian R Gaines

Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4
gaines@calgary.cdn

ABSTRACT

Knowledge acquisition has developed from manual techniques in the 1970s through a first generation of computer-based tools in the early 1980s to a second generation of integrated knowledge support systems in the late 1980s. This paper reviews the requirements for such second generation systems, places the integration requirements within the framework of support for human knowledge processes, analyses the theoretical foundations of such systems through a model of knowledge validation, and analyses the technical foundations of such system through a model of generations of information technology. It concludes with an overview of the systems architecture of second generation knowledge acquisition systems.

INTRODUCTION

The development of knowledge acquisition systems is facing a critical transition. The zeroth generation of knowledge acquisition techniques in the 1970s were based on human interviewers debriefing experts about their skills. The first generation of computer-based tools in the 1980s were based on the computer-aided application of specific techniques such as construct and conceptual mapping, behavior modeling and text analysis. However, neither manual nor tool-based approaches have yet made a significant impact on the knowledge acquisition bottleneck which continues to impede the development and effective application of knowledge-based systems. Current, second generation research is establishing a new paradigm for knowledge acquisition, one that recognizes the wide diversity of forms and sources of knowledge, its essential dynamism making it subject to contradiction and change, and the constructive nature of knowledge acquisition whereby it is often the knowledge acquisition process itself that creates knowledge.

Second generation knowledge acquisition targeted on applications in the 1990s have to:

- Integrate a wide variety of weak methods encompassing the diversity of forms and sources of knowledge
- Integrate conflicting sources of knowledge to cope with essential contradictions that cannot be replaced by an artificial and misleading 'consensus'
- Integrate knowledge acquisition with system operation to facilitate a heavy activity of continuing system maintenance and upgrade as knowledge changes
- Integrate knowledge representation in the acquisition and performance systems, retaining the flexibility for new representations to be used that more accurately carry the knowledge

Thus, second generation developments have two objectives that may come into conflict (Gaines 1987):

- To integrate closely a variety of techniques and technologies to cope with the diversity of knowledge
- To provide flexibility within the integration framework for new techniques and technologies to be incorporated to cope with the dynamics of knowledge

This paper analyses the integration requirements within the framework of support for human knowledge processes, analyses the theoretical foundations of such systems through a model of

knowledge validation, analyses the technical foundations of such system through a model of generations of information technology, and uses these analyses to develop an architecture for second generation developments.

THE NATURE OF KNOWLEDGE

The diversity of human knowledge processes cannot be over-emphasized—indeed 'anarchy' might be a better term than diversity (Feyerabend 1975). "Knowledge" in the artificial intelligence literature has become associated with formal structures from which inferences can be made. However, much human knowledge is not well-structured for formal representation and computer processing (Gaines 1986). The philosophical dictionary definition of knowledge as 'more than opinion, less than truth' (Runes 1972) encapsulates what can be said. Classical data processing methods take it for granted that the data can be assumed to be 'true' and base their inferences on this. In knowledge-based system we have data that is less than truth, but more than opinion because it is 'expert opinion'. The implications of these notions of untruth (inconsistency, contradiction, change) and expertise (modeling, cognitive authority, reflective equilibrium) are critical to the design of knowledge acquisition systems.

It is interesting to note that in the late 1980s computer-based systems have been developed that are targeted on the representation of very loosely structured knowledge with less emphasis on formal structures from which inferences can be made. *Hypertext* and *hypermedia* systems approach knowledge a different direction to that of expert systems, accepting it as having limited structure and processibility, and adding a limited structure through the links that is intended to aid the user rather than allow computer-based inferencing (Conklin 1986, Gaines & Vickers 1988). They extend knowledge representation to include any media whereby human knowledge can be transferred even if there is no means for analyzing that knowledge or its transfer. For example, a film of a skilled person performing a task which enables another person to perform that task better contains knowledge which can be accessed and transferred through a hypermedia system (Vickers & Kingston 1987).

Hypermedia weaken the formal, computational nature of the current concepts of 'knowledge bases' (Brodie & Mylopoulos 1986) in not requiring that the information system be able to represent and process the knowledge encoded in the media used. However, if the links in hypertext systems are labeled (Trigg 1983) then the possibility of automatic inferencing is enhanced, and if arbitrary processes can be associated with the traversal of links (HyperCard 1987) then the boundary between user-mediated inferencing and computer-mediated inferencing becomes blurred.

The notions of computer-based hypermedia generate the interesting conceptual framework that a knowledge-based system is a medium for carrying human knowledge, and hence its functionality and operation can be related to those of existing media. Figure 1 shows how this framework can be developed to relate human knowledge processes with the relevant techniques used, the pre-computer supporting technology, and the computer-based supporting technology.

The basic level is that of human *experience* as the generative substratum for knowledge processes. In the survival of the species it is important that experience and the lessons learnt from it are shared. Others may be able to learn from experiences more effectively than those who have them. Individuals are finite and expendable but their experience is part of the knowledge economy of the species. Hence the next level is that of *communication* of experience.

However, the process of communication goes beyond sharing of experience because it allows for indirect action upon the world by persuading others to act in one's place, and this needs to be differentiated as a different form of communication, that of *discourse*. Discourse involves the use of communication intentionally to affect the behavior of others and, in particular, to communicate not just experience but *understanding*.

	Relevant Techniques	*Supporting Technology*	*Computing Technology*
Experience	Perception, Action	Instruments, Tools	Games, Simulation
Communication	Mimicry, Representation, Language	Painting, Writing, Phonograph, Television	Word Processing, Graphics, Hypermedia
Discourse	Rhetoric, Communicative Action	Books, Newspapers, Journals, Telephone	Electronic Mail, Structured Hypermedia
Practical Reasoning	Logic, Induction, Analogy	Jurisprudence Systems, Uncodified 'Commonsense'	Semantic Nets, Conceptual Graphs
Formal Reasoning	Formal Logic	Predicate Calculus, Sequent Calculi	Relational Databases, Expert Systems
Mathematical Reasoning	Set Theory, Category Theory	Calculators, Metatheorems, Proof Techniques	Number Crunchers, Theorem Provers

Fig.1 Knowledge processes, techniques and support

The next level of *practical reasoning* is concerned with the creation of knowledge from experience, and its application to the generation of experience—that is, determination of the meaning, significance, consequences, actions and principles based on experience.

The next two levels, of *formal* and *mathematical* reasoning, are separated from practical reasoning because of their fundamental role in the development of modern science and technology, and their significance in computing. Formal reasoning provides foundations for precise deductive processes and mathematical reasoning applies this to the structures underlying arithmetic and algebraic systems.

The techniques and support systems underlying these knowledge processes are shown in three columns in Figure 1:

• Experience
—Relevant Techniques: human interaction with worlds through perception and action
—Supporting Technology: instruments for enhancing perception and tools for enhancing action
—Computing Technology: the creation of artificial experience through games and simulation
• Communication
—Relevant Techniques: demonstration and mimicry of behavior, representation of experience through media, description of experience through language
—Supporting Technology: painting, writing, phonograph and television as media for capturing experience
—Computing Technology: word processing, graphics and hypermedia as media for representing and presenting experience
• Discourse
—Relevant Techniques: rhetorical techniques for persuasion and communicative action to secure agreement over an understanding
—Supporting Technology: rhetorical and interactive media such as books, newspapers, journals and the telephone
—Computing Technology: electronic mail supporting conversation and structured hypermedia supporting the presentation of arguments
• Practical Reasoning
—Relevant Techniques: informal deductive logic, inductive and abductive inference, and analogical pattern recognition
—Supporting Technology: surprisingly little—jurisprudence through the legal system to support justice in socially important practical reasoning, but primarily left to uncodified 'commonsense'
—Computing Technology: knowledge representation and inference through semantic nets and conceptual graphs
• Formal Reasoning
—Relevant Techniques: formal logic based on axiom systems
—Supporting Technology: various calculi for formal logics such as predicate and sequent calculi
—Computing Technology: relational databases and expert systems
• Mathematical Reasoning
—Relevant Techniques: set theory and category theory
—Supporting Technology: calculators for arithmetic, metatheorems and proof techniques for algebraic systems
—Computing Technology: number crunchers and spreadsheets for arithmetic, theorem provers for algebraic systems

Looking down the right hand column of Figure 1 we can see the full spectrum of computer-based techniques supporting human knowledge processes, and the need for integration becomes very apparent. A knowledge support system must be capable of providing any combination of graphics, text processing, hypermedia, number crunching, databases, communications, theorem proving, databases, and semantic nets and expert system technologies. These are not highly separated functions but rather multiple aspects of human knowledge processing.

It is particularly interesting to note the weaknesses in the support of discourse and practical reasoning processes. These are the levels at which the colloquial concept of 'expertise' is highly applicable. However, expert systems technology has developed primarily at the level of formal reasoning which is a very narrow subdomain of human reasoning processes.

THE VALIDATION OF KNOWLEDGE

The analysis of the previous section may be taken a stage further by considering the processes by which knowledge can be validated, that is rationally developed from the realm of opinion towards that of truth. Typical criteria used for distinguishing knowledge from opinions are: observation; reliable evidence; rigorous application of method; the use of a precise language; testing by experiment; power of anticipation; formalization; ability to proceed to applications; usefulness of these applications; accountability to certain principles or to acquired knowledge; impartiality; accepted views of specialists (Franck 1979).

These criteria can be portrayed systematically using Popper's (1968) distinctions between three 'worlds': "If we call the world of things—of physical objects—the first world and the world of subjective experience the second world we may call the world of statements in themselves the third world (world 3)." This definition can be expanded in terms of the distinctions involved:

- World 1 arises from the assertion that there exist *physically necessary* distinctions

—this defines a world of *physical systems*

—a key concept is that of *actuality*, that the distinctions made are actual

—the conceptual framework involved is that of *epistemology*, that the distinctions have to be drawn from the actual world

—truth in this world is by *correspondence*, that the distinctions made correspond to the properties of physical systems

—inference is *causal*, based on chains of physical cause and effect

- World 2 arises from the assertion that there exist *humanly chosen* distinctions

—this defines a world of *mental systems*

—a key concept is that of *agency*, that distinctions have a maker

—the conceptual frameworks involved are those of *psychology and sociology*, that distinctions exist in the mental worlds of individuals and groups

—truth in this world is *performative*, that the distinctions chosen will be continue to be chosen

—inference is *conventionalist*, based on accepted chains of reasoning that are chosen within a culture

- World 3 arises from the assertion that there exist *formal systems* of distinctions

—this defines a world of *coherent systems*

—a key concept is that of *abstraction*, that systems of distinctions have their own existence

—the conceptual framework involved is that of *ontology*, that some distinctions are made independent of any criteria other than intrinsic structure

—truth in this world is by *coherence*, that the systems of distinctions have an acceptable intrinsic structure

—inference is *structuralist*, based on chains of reasoning that conform to the structure

Figure 2 shows the interactions between a knowledge-based systems and these three worlds in ways that may be used to validate it:

- *Prediction* is the capability to use knowledge to model the world and predict its behavior
- *Action* is the capability to use knowledge to effectively control the world
- *Explanation* is the capability to use knowledge to explain the basis of predictions and actions
- *Understanding* is the capability to accept new knowledge through discourse
- *Derivation* is the capability to deduce knowledge from coherent knowledge systems
- *Creation* is the capability to add knowledge to coherent knowledge systems

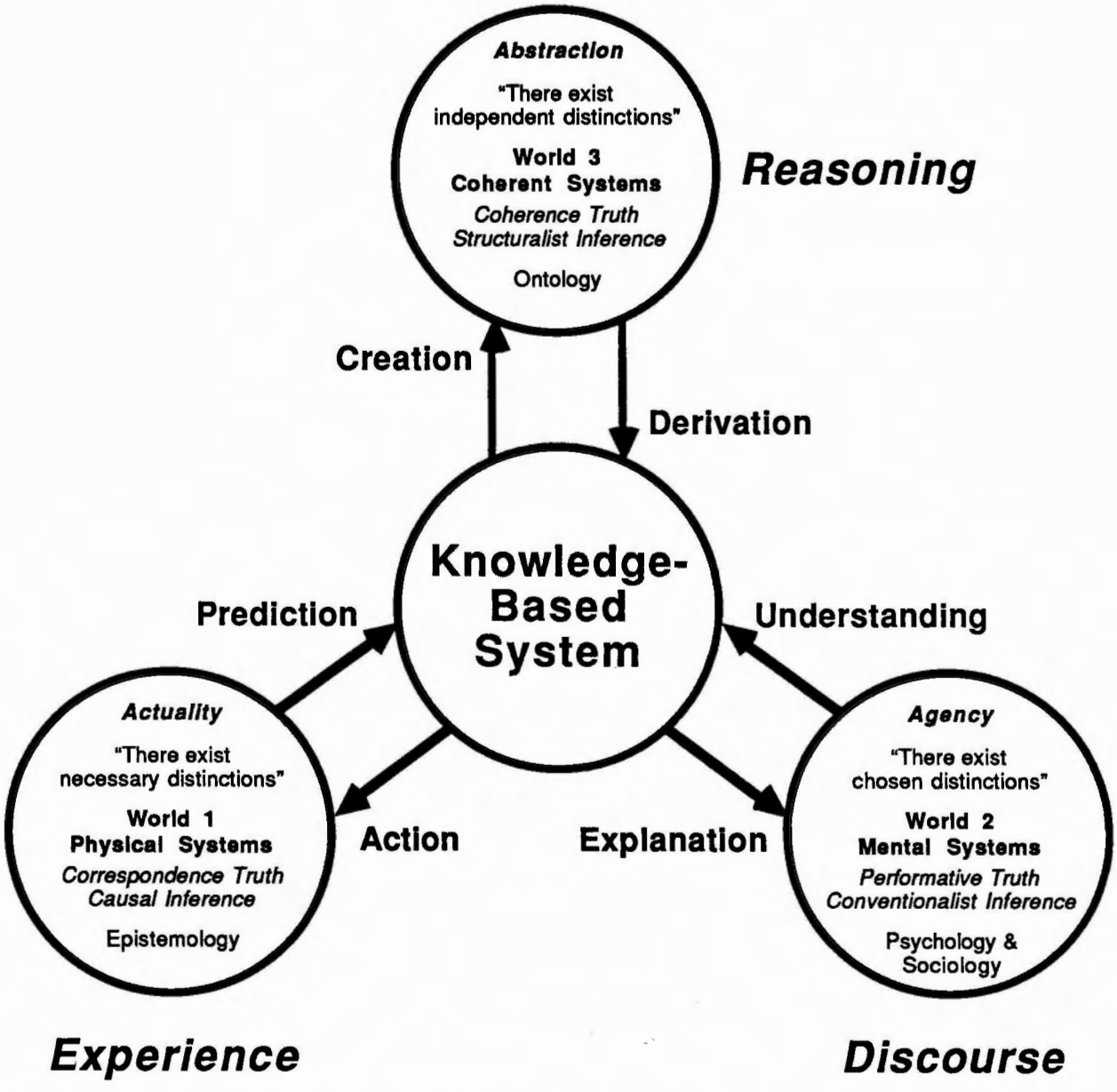


Fig.2 Interaction of a knowledge-based system with 3 worlds

For example, the MYCIN/TEIRESIAS system (Shortliffe 1976, Davis & Lenat 1982) exhibits all these activities to some limited extent. It predicts microbial infections from clinical data and recommends appropriate action. It gives explanations of these predictions and is able to understand new knowledge given in natural language. It derives its knowledge from a coherent set of rules and is able to create new rules from linguistic input. Analysis of the strengths and weaknesses of this and related systems in these various areas has been the major source of progress in the development of knowledge-based systems (Hasling, Clancey & Rennels 1984, Chandrasekaran & Mittal 1984).

Figure 2 also illustrates how the 3 worlds may be used to give foundations to the distinctions between levels in Figure 1, with experience in world 1, communication and discourse in world 2, and reasoning in world 3. The 3 worlds model provides a theoretical framework for the development of second generation integrated knowledge acquisition systems.

It is apparent in Figure 2 that the links from the knowledge-based system to world 1 have been studied in depth in the methodologies and philosophies of science. The links to world 3 have been studied in depth in the methodologies and philosophies of deductive and inductive inference. Both these domains of study have been used as major input to the development of computer-based systems. However, the links to world 2 which are studied in depth in the psychology, sociology and philosophy of discourse are the weakest in current knowledge-based systems. Explanation and understanding will be crucial areas of development in second generation knowledge acquisition systems.

The sociological basis of explanation and understanding has been developed in depth by Habermas (1984) through his concept of *communicative action*. Habermas distinguishes four primary components of communication acts: *strategic* in which one person treats another as an entity whose behavior is manipulable through language; *normative* in which the communication conforms to shared norms and values; *dramaturgical* in which one person presents their self to others; and *communicative* in which people attempt to reach voluntary agreement on their situation for the sake of cooperating. Habermas argues for communicative action as the foundation of a healthy society.

One can characterize the move towards knowledge-based systems in computing as being one of developing the possibilities for communicative action by basing the systems on overt, high-level data structures. 'Overt' means that the structures are separate from, and not embedded in the technical code of the computer. 'High-level' means that they are understandable by non-computer specialists having a knowledge of the relevant knowledge domain. Communicative action is the capability to provide understandable explanations of data-processing using these overt, high-level structures, and to understand specifications of data-processing in them.

Popper's concept of world 3 is particularly interesting from the viewpoint of explanation and understanding since he sees world 3 as one of *objective knowledge* where the primary criterion is one of *fallibilism*—there must be tests that might show the knowledge to be false. However, this criterion which establishes the possibility of a critical community, also establishes the need for knowledge to be *understandable* so that it can be tested by the community (Notturno 1985). This is consistent with Habermas' analysis of communicative understanding provided the distinction between truth and falsity is taken to include validation in worlds 2 and 3 as well as world 1.

Pask (1980) has developed a similar framework to Habermas as a foundation for computer-based learning. His *conversation theory* singles out as the critical event in discourse the occasions on which two participants agree that they have effectively exchanged a concept and validated its mutual similarity. Such *agreement over an understanding* punctuates the discourse into *strict conversations* whose structure and processes can be analyzed in depth as coherent units. Pask recognizes the diversity of validity tests that can be applied to understanding such as the ability to: *produce* knowledge transferred; to *reproduce* knowledge transferred from an alternative derivation; to *apply* knowledge transferred to problem solving; and, recursively, to *teach back* knowledge transferred.

The significant common theme in Habermas', Popper's and Pask's analyses is that the discourse should provide for as much exploration of a knowledge structure and its validity as the parties to it require in order to establish agreement or disagreement. For a knowledge-based system this implies the capability to expose links over as wide a breadth and in as much depth as the user requires. For knowledge acquisition systems this implies that a minimal translation of expert skills into computational form is inadequate. The acquisition phase has to be able to explore the full richness of the knowledge structure encompassing linkages which, although not computationally useful, may be vital to explanation and understanding.

GENERATIONS OF INFORMATION TECHNOLOGY

To examine the technological support for second generation integrated knowledge acquisition systems it is necessary to have some model of the wide range of technologies underlying computing. The computer itself is only one component in an elaborate and highly differentiated infrastructure which has grown through a succession of generations of computers, each of which represents a major change in computing technology (Gaines 1984). During the eight year span of each computing generation, revolutionary changes have taken place that correspond in magnitude to those taking place over some hundred years or more in other industries. The first generation commenced in 1948 and we are now in the sixth generation spanning 1988 through 1995.

The generation structure of information technology can itself be analyzed in terms of the knowledge flows involved. Like other basic technologies (Marchetti 1981), the growth of computing has an infrastructure comprised of the envelope of learning curves in successive underlying technologies (Ayres 1968). Figure 3 shows this structure through the zeroth through fifth generations projected to the sixth and seventh. Each technology has a learning curve in which a *breakthrough* leads to a phase of: *replication* in which the results are duplicated; *empiricism* in which rules for design are derived from experience; *theory* in which generative principles are derived; *automation* in which the theory is made operational; and *maturity* when the knowledge becomes proven, widely available and widely used (Gaines & Shaw 1986).

At the base is the learning curve for different electronic device technologies: 1940 zeroth generation using relays; 1948 first using vacuum tubes; 1956 second using transistors; 1964 third generation using using integrated circuits; 1972 fourth using large-scale integration; 1980 fifth using VLSI; 1988 sixth projected to use ultra large-scale integration with some 10 million transistors on a chip; 1996 seventh projected to use grand-scale integration with 1,000 million on a chip. The definition of generations in terms of electronic device technology captures some important aspects of computing such as cost decreases, size decreases, power increases, and so on. However, it fails to account for the qualitative changes that have given computing its distinct character in each generation. These appear through the tiered succession of learning curves of higher level technologies based on the lower level developments.

Now the generations may be defined in terms of major qualitative changes:

- the initial breakthrough in *electronic device technology* leading to the zeroth generation is placed at 1940 about the time of the Atanasoff and Berry experiments with tube-based digital calculations
- the first generation breakthrough was the introduction by Mauchly and Von Neumann of stored program and subroutine concepts around 1947 which led to the transition from the ENIAC to the EDVAC designs—this detached computing as a separate discipline from electronics by substituting software for hardware in a *virtual machine architecture*
- the second generation breakthrough was to bridge the gap between machine and task through the development of *problem-oriented languages* such as FORTRAN around 1956
- the third generation breakthrough was to bridge the gap between the computer and the person with the development of interactive time-shared computers in 1964 allowing close *human-computer interaction*
- the fourth generation breakthrough was in the early 1970s with developments in expert systems based on knowledge-processing—such *knowledge-based systems* have the capability to store information through its inter-relations and make inferences about its consequence
- the fifth generation breakthrough was in 1980 with developments in machine learning and *inductive inference systems*

- the growth of robotics will provide the next breakthroughs in which goal-directed, mobile computational systems will act autonomously to achieve their objectives—the breakthrough into the sixth generation era commencing in 1988 will be one of *autonomous activity systems*
- the interaction between these systems will become increasingly important in enabling them to cooperate to achieve goals—the seventh generation era commencing in 1996 will be one of *socially organized systems*

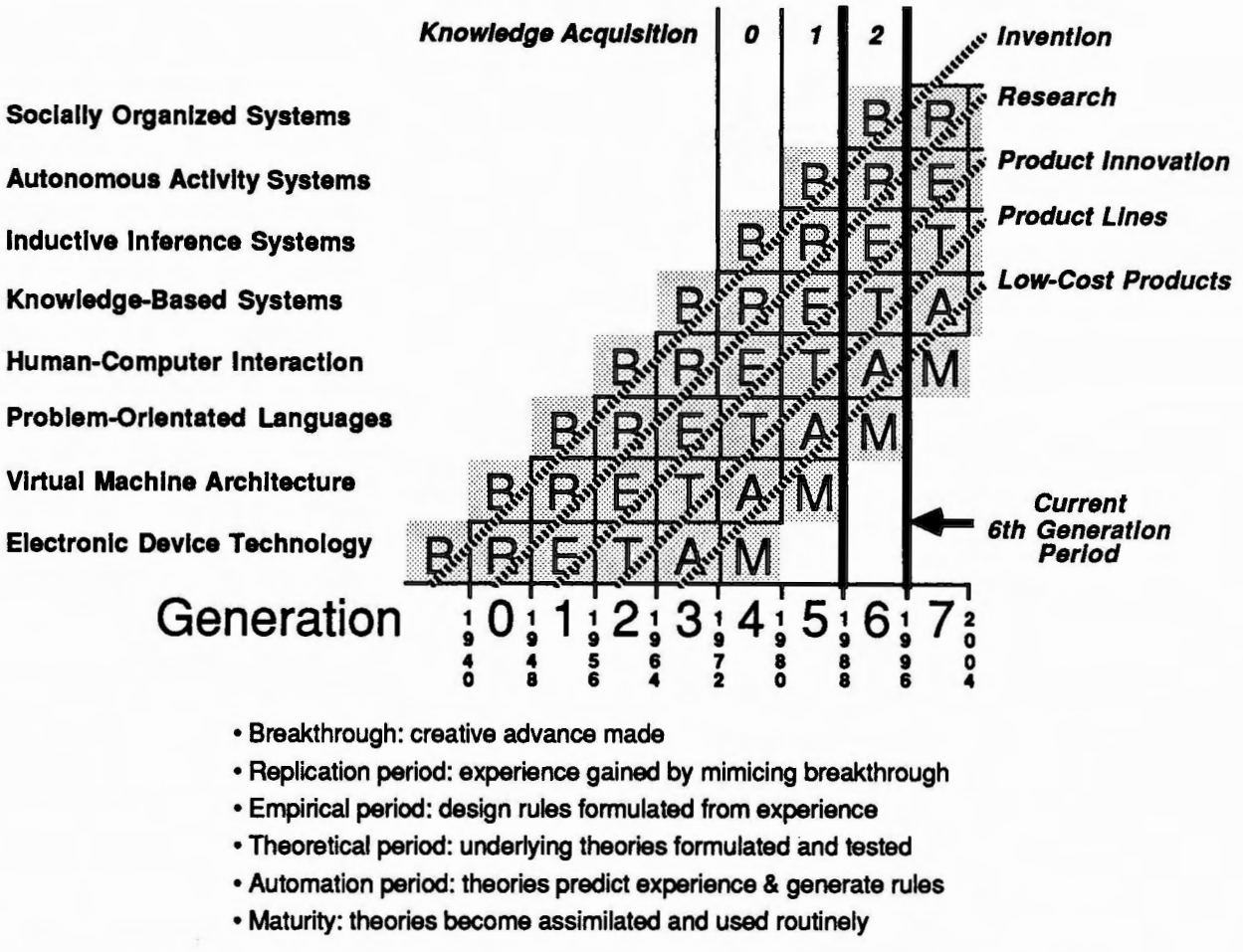


Fig.3 Knowledge acquisition in the Infrastructure of Information Technology

The model of the development of computing described above is important in any evaluation of the impact of computing. Each generation represents a revolution in technology with a qualitatively different impact. Each generation subsumes the capabilities of that preceding it, providing very much better facilities at very much lower cost, and adding new capabilities not possessed by the previous generations. The availability of these new capabilities in a generation is linked to the stage of the learning curve for each technology in that generation as shown in Figure 3:

- *Invention* is focused at the **BR** interface where new breakthrough attempts are being made based on experience with the replicated breakthroughs of the technology below;
- *Research* is focused at the **RE** interface where new recognized breakthroughs are being investigated using the empirical design rules of the technologies below;
- *Product Innovation* is focused at the **ET** interface where new products are being developed based on the empirical design rules of one technology and the theoretical foundations of the technologies below;

- *Product Lines* are focused at the **TA** interface where product lines can rest on the solid theoretical foundations of one technology and the automation of the technologies below;
- *Low-cost Products* are focused at the **AM** interface where cost reduction can be based on the the automated mass production of one technology and the mature technologies below.

For example, in the fourth generation (1972-79):

- **BR**: recognition of the knowledge acquisition possibilities of knowledge-based systems led to the breakthrough to inductive-inference systems;
- **RE**: research focused on the natural representation of knowledge through the development of human-computer interaction, e.g. the Xerox Star direct manipulation of objects;
- **ET**: experience with the human-computer interaction using the problem-oriented language BASIC led to the innovative product of the Apple II personal computer;
- **TA**: the simplicity of the problem-oriented language RPG II led to the design of the IBM System/3 product line of small business computers;
- **AM**: the design of special-purpose chips allowed the mass-production of low-cost, high-quality calculators.

In the fifth generation (1980-87):

- **BR**: recognition of the goal-seeking possibilities of inductive inference systems is leading to the breakthrough to autonomous-activity systems in robotics;
- **RE**: research is focused on learning in knowledge-based systems;
- **ET**: the advantages of the non-procedural representation of knowledge for human-computer interaction led to the innovative designs of the Visicalc spread-sheet business product and the Lisp-machine scientific product;
- **TA**: the ease of human-computer interaction through a direct manipulation problem-oriented language led to the Apple Lisa/Macintosh product line of personal computers;
- **AM**: the design of highly-integrated language systems has allowed the mass-production of low-cost, high-quality software such as Turbo Pascal.

The generations of knowledge acquisition systems are shown at the top of Figure 3, and the current sixth generation of information technology is also marked. It coincides with the second generation of knowledge acquisition systems and this four generation gap indicates an analogy between the state of knowledge-based systems development now and that of data processing 32 years ago. We do indeed have a similar situation to the 1950s in computing when software development was new with problems of shortage of skills, unrealistic schedules and budget over-runs.

The detailed infrastructure for the development of the second generation of knowledge acquisition systems may be seen in Figure 3. At the three lowest levels, the hardware machines architectures and languages are all mature technologies posing no problems. Human-computer interaction has entered its phase of automation with the increasing availability of machine-independent interactive systems such as X-windows. Knowledge-based systems have entered the theoretical development phase with increasing depth and structure to knowledge representation and inferencing schema. Inductive inference systems have entered the empirical phase where there are beginning to be available a range of usable tools. Autonomous activity systems are still in the research phase and the lack of experience and foundations will restrict the development of knowledge acquisition systems for goal-directed, planning activities during this generation.

Thus overall, we should expect the emphasis in second generation knowledge acquisition systems to be on the use of high-speed graphic workstations, programmed in high-level languages, using high-quality user interfaces to provide effective knowledge representation, some inductive inferences and little or no capability to deal with planning.

A SYSTEMS ARCHITECTURE FOR THE SECOND GENERATION

A general framework for a systems architecture for the second generation of knowledge acquisition systems may be developed based on the analysis of the previous three sections.

1 The distinction between the acquisition phase and the performance phase will begin to break down so that one is talking of a generation of *knowledge support systems* allowing users to switch modes between being knowledge sources and knowledge appliers.

This lack of clear role definition in second generation systems has other interesting consequences since it merges the concepts of knowledge acquisition and computer-assisted learning, distinguished only by the intended direction of flow of knowledge. Figure 4 shows this in terms of the knowledge flows in a knowledge support system. Clearly, each flow involves "agreement over an understanding" and hence two-way interaction. Clearly, also, the three roles shown can be within one individual acting sometimes to add knowledge to the system, sometimes to take its advice and sometimes to learn from it.

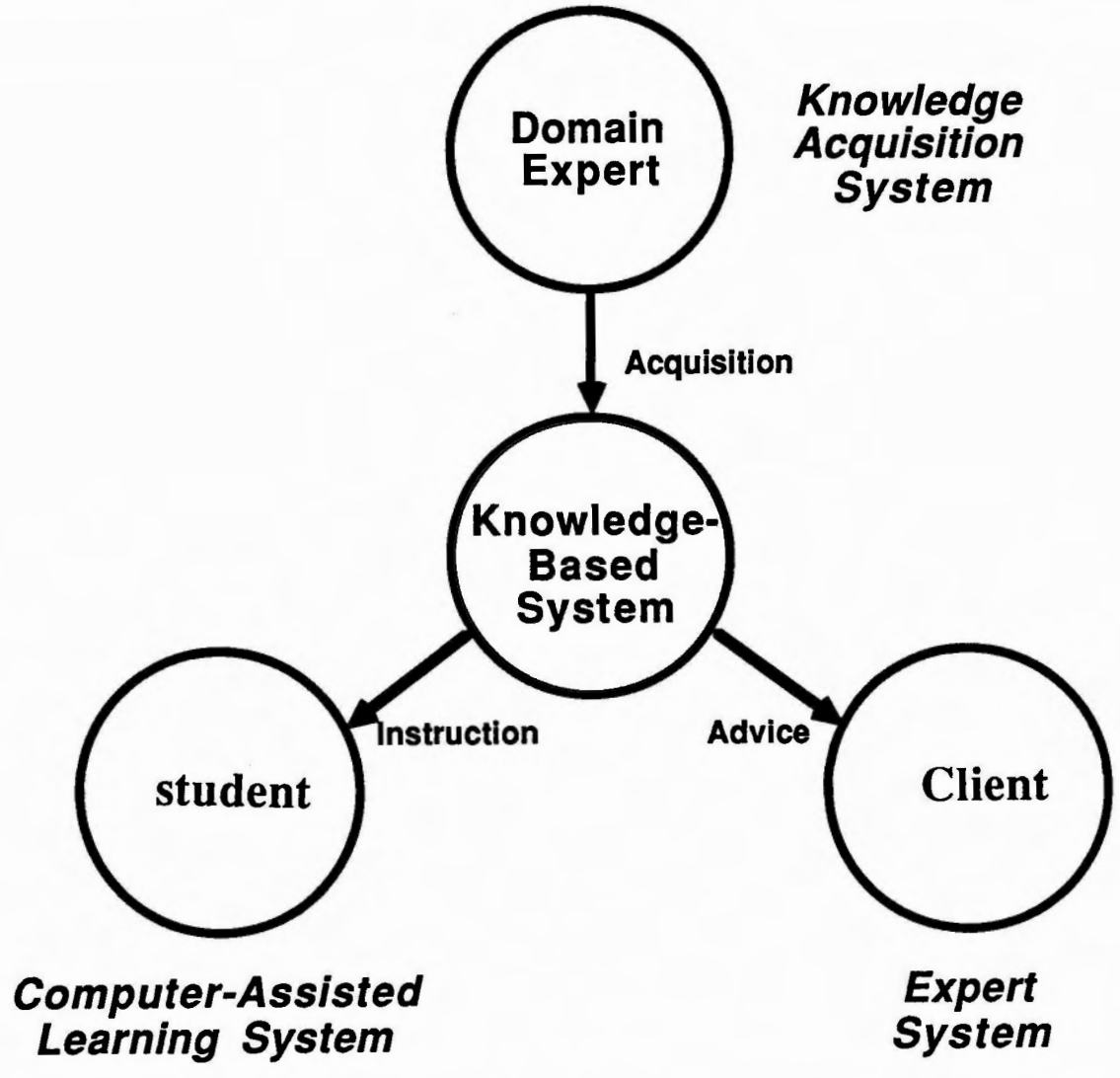


Fig.4 Knowledge flows in a knowledge support system

2 Acquisition will not be dominated by the requirement for computer-based inferencing but will encompass knowledge that can be interpreted only by the user as well as that interpretable by the computer.

In particular, acquisition will be as much for explanation and understanding as it is for inferencing.

3 A basic data item in knowledge support systems will be an uninterpreted knowledge representation objects encompassing text, graphics, sound, videodisc sequences, and so on.

The capability for objects to establish their identity, meaning, significance, denotation, and so on through direct interaction with the user is very important. The computer-based system does not have to understand the processes involved, only that they may have occurred.

4 Hypermedia-like links will be added to these data objects, some of which will be only user interpretable, but others of which will be computer-interpretable for inferencing.

This structure of objects, attributes and relations encompasses hypertext, semantic networks, conceptual graphs, and repertory grids, among other possibilities.

5 Some links will have associated validation procedures (methods) that test the validity of a link in terms of associated structures and hence may also be used to derive such structures from the link.

This association of validity tests with links allows the introduction of abstraction hierarchies and inference rules within the object-link framework.

The notion of a validity test is recursive since it introduces further knowledge within the system which itself will be presented by objects and links and subject to validity tests. In practical systems the infinite regression possible will be truncated at whatever level is acceptable to users. Further knowledge acquisition will be performed as acceptance criteria change.

6 A major part of the system will be devoted to link integrity maintenance which will be the prime dynamic to the generation and control of knowledge within the system.

In particular, structures when edited will reflect changes into associated structures. This is a weakness in most current knowledge acquisition schema in that they are unidirectional in deriving computational knowledge from expert data and do not reflect edits in the knowledge base back to the knowledge sources.

7 The hardware architecture will be a network of high speed graphic workstations connected through communications networks and utilizing database and computational servers.

In most cases the hardware will be fairly heterogeneous.

8 The software architecture will be based on an object-oriented language, with objects dynamic at run-time, providing the overall system control.

This will be able to invoke processes in other languages, however, so that the overall software architecture will also be heterogeneous.

None of these requirements is technically infeasible currently. Second generation knowledge acquisition systems are possible with existing hardware, software and knowledge technologies.

ACKNOWLEDGEMENTS

I am grateful to many colleagues for discussions over the years that have influenced this paper. In particular I would like to thank John Boose, Jeff Bradshaw, Bruce Conrad, Debbi Leishman, Gordon Pask, Mildred Shaw and Joan Vickers, for access to their own research and many stimulating discussions.

REFERENCES

Ayres, R.U. (1968). Envelope Curve Forecasting. **Technological Forecasting for Industry and Government.** pp. 77-94. New Jersey: Prentice Hall.

Brodie, M.L. & Mylopoulos, J. (Eds) (1986). **On Knowledge Base Management Systems.** New York: Springer Verlag.

Chandrasekaran, B. & Mittal, S. (1984). Deep versus compiled knowledge approaches to diagnostic problem solving. Coombs, M.J. (Ed). **Developments in Expert Systems.** pp.23-34. London: Academic Press.

Conklin, J. (1986). A survey of hypertext. MCC Technical Report STP-356-86. Austin:Microelectronics & Computer Technology Corporation (October).

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

Feyerabend, P. (1975). **Against Method.** London: NLB.

Franck, R. (1979). Knowledge and opinions. Nowotny, H. & Rose, H. (Eds). **Counter-movements in the sciences.** pp.-. Dordrecht, Holland: Reidel.

Gaines, B.R. (1984). Perspectives on fifth generation computing. **Oxford Surveys in Information Technology, 1,** 1-53.

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.

Gaines, B.R. (1987) Advanced expert system support environments. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.8-0-8-14. Banff (October).

Gaines, B.R. & Shaw, M.L.G. (1986). A learning model for forecasting the future of information technology. **Future Computing Systems, 1**(1), 31-69.

Gaines, B.R. & Vickers, J.N. (1988). Design considerations for hypermedia systems. **Microcomputers for Information Management,** 5(1), 1-27 (March).

Habermas, J. (1981). **The Theory of Communicative Action: Reason and the Rationalization of Society.** Boston: Beacon Press.

Hasling, D.W., Clancey, W.J. & Rennels, G. (1984). Strategic explanations for a diagnostic consulting system. Coombs, M.J. (Ed). **Developments in Expert Systems.** pp.117-133. London: Academic Press.

HyperCard (1987) **HyperCard User's Guide.** Cupertino, California: Apple Computer.

Marchetti, C. (1981). Society as a learning system: discovery, invention and innovation cycles revisited. Report RR-81-29 (November). Laxenburg, Austria: International Institute for Applied Systems Analysis.

Notturno, M.A. (1985). **Objectivity, Rationality and the Third Realm: Justification and the Grounds of Psychologism: A Study of Frege and Popper.** Dordrecht, Holland: Martinus Nijhoff.

Pask, G. (1980). Developments in conversation theory—Part I. **International Journal of Man-Machine Studies, 13**(4), 357-411 (November).

Popper, K.R. (1968). Epistemology without a knowing subject. Van Rootselaar, B., Ed. **Logic, Methodology and Philosophy of Science III**. pp. 333-373. Amsterdam, Holland: North-Holland Publishing Co.

Runes, D.D. (1972). **Dictionary of Philosophy**. London: Peter Owen.

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

Trigg, R.H. (1983). **A Network-Based Approach to Text Handling for the Online Scientific Community**. PhD Thesis, University of Maryland (University Microfilms 8429934).

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.

A NEW BREED OF KNOWLEDGE ACQUISITION SYSTEMS USES REDUNDANT KNOWLEDGE?!

M. Gams
Jozef Stefan Institute
Computer Science Department
Artificial Intelligence Laboratory
Jamova 39
61000 Ljubljana
Yugoslavia

tel.: (+38)(61)214-399
tlx.: 31-296 YU JOSTIN

ABSTRACT. Most of expert systems are working in domains where the importance of appropriate mechanisms for knowledge acquisition is largely enhanced. Real-world domains are characterized by imperfect data and imperfect (probabilistic) rules. Designers of latest systems for knowledge acquisition have devoted most of their attention to special procedures for dealing with noise. Research in empirical learning provided better theoretical and practical understanding of the whole process. The new breed of knowledge acquisition systems take the best of both worlds: statistics and artificial intelligence, provide explanation in a human understandable way and achieve better classification accuracy than classical statistical methods. Another possibility is becoming feasible - to construct knowledge acquisition tools which represent specific families of algorithms in one unifying algorithm.

GINESYS is a partial implementation of the unifying algorithm for a family of empirical learning systems. A new mechanism is presented - confirmation rules, a form of redundant knowledge. Empirical tests were made with GINESYS with different parameters on two real-life oncological domains. Results indicate that the use of redundant knowledge enables significant improvements in classification accuracy approaching the upper limit in noisy domains.

Keywords: knowledge acquisition, machine learning, concept learning, empirical techniques, theoretical analysis.

1. INTRODUCTION

Empirical learning (EL) (Michalski et al. 1983) or 'structural descriptions learning' (Forsyth & Rada 1986) denotes learning from examples. Its aim is to induce general descriptions of concepts from instances (examples) of these concepts. Usually examples are objects of a known class described in terms of attributes and values. The final product of learning are symbolic descriptions expressed in high-level, human oriented terms and forms. Induced descriptions of concepts, representing different classes of objects, can be used for classifying new objects. They can be viewed as a new representation of the domain knowledge.

EL systems are knowledge acquisition tools. In this task the communication with a user is of greatest importance (Michalski et al. 1983; Kodratoff 1987; Breiman et al. 1984; Lavrac et al. 1986). Basic advantages of AI over classical methods are (a) explanation during classification of a new example and (b) explanation of the whole problem domain. The ability (a) is

obtained by tracing the classification rules and (b) by constructing the knowledge base.

Some EL systems achieved about the same classification accuracy as statistical methods (Breiman et al. 1984; Michalski & Chilausky 1980; Michalski et al. 1986; Forsyth & Rada 1986; Kononenko 1985; Gams 1987; Gams & Petkovsek 1988).

2. GINESYS

Some well known algorithms (C4 (Quinlan 1986), CART (Breiman et al. 1984), ASSISTANT (Bratko & Kononenko 1987), ASSISTANT 86 (Cestnik et al. 1987), CN2 (Clark & Niblett 1987), AQ15 (Michalski et al. 1983)) were reviewed in detail and a general schemata was proposed (Gams & Lavrac 1987), resulting in the design of a unifying algorithm. The existing partial implementation of the proposed unifying algorithm UAEL (Unifying Algorithm for Empirical Learning) is GINESYS (Generic INductive Expert SYstem Shell). Although GINESYS enables only generation of rules in a top-down mode of UAEL, it enables choosing between many parameters, such as different impurity functions, error estimates, forms of the constructed classification rule etc.

New mechanisms such as 'confirmation rules' (the use of redundant knowledge) and a new algorithm for cross-validation (Watkins 1987; Breiman et al. 1984; Niblett 1987) are introduced. Here we concentrate on the confirmation rules.

3. CONFIRMATION RULES

The idea of confirmation rules comes from systems like DENDRAL (Buchanan & Feigenbaum 1978) where redundant knowledge (different sources of information) was used with a great success. Basic principle of using redundant knowledge in EL systems can be shown by a simple analogy: instead of one expert (one rule), several experts (rules) are consulted. Classification is unquestionable when all experts (rules) agree. If significant disagreement is observed, a probabilistic method is taken for a classical conflict resolution (Waterman & Hayes-Roth 1977). It seems that the use of redundant knowledge or using several sources of information is a common practice in every-day life.

A classification rule in most existing empirical learning systems has the following form:

```
if Cond1 then Class1
   else if Cond2 then Class2
```

Each row corresponds to one rule or a branch in the tree. Confirmation rules are added to each of the rules in previous form:

```
if Cond1  then Class1

  (Cond11 then Class11
   ..................
   Cond1R then Class1R)

   else if Cond2  then Class2

          (Cond21 then Class21
          ..................
           Cond2R then Class2R)
```

Classification with confirmation rules starts by checking the first rule at each level in a top-down way until at some level

the first rule succeeds. In that case confirmation rules are checked. Confirmations rules that match a classified example, add their class probability distribution to the summary distribution according to the formula for the union of probabilities (p12 = p1 + (1-p1)*p2). The probabilities are multiplied by weights (=error estimates). If the final classification distribution is 'bad' according to a predefined criteria, a Bayesian classifier is consulted as a method of conflict resolution.

The explanation during classification with confirmation rules can be minimal - only with the first rule when it succeeds (or with a warning when conflict occurs). Maximal explanation displays all possible information.

The informativity of the knowledge base with confirmation rules seems to be much better than without them. If needed, confirmation rules can be trimmed off and a 'usual' knowledge base is obtained as a downgraded version.

3.1. A SIMPLE EXAMPLE

In the following example we are trying to learn who can born. We have two attributes (dress, sex) and two classes (yes, no).

DRESS	SEX	!	CAN_BORN
SKIRT	FEMALE	!	YES
TROUSERS	MALE	!	NO

Given these two examples most of the existing systems would construct the following rules:

```
if dress=skirt then can_born=yes
  else (if dress=trousers) than can_born=no
```

(Some systems construct the condition in the parentheses and some not.) GINESYS with confirmation rules constructs the following

```
if dress=skirt then can_born=yes
   dress=trousers then can_born=no
   sex=female then can_born=yes
   sex=male than can_born=no
  else can_born=no
```

and when classifying for example a Scot (dress=skirt & sex=male) answers with can_born=yes:no with a 50:50 % probability. Note that prediction without confirmation rules would be can_born=yes.

4. EMPIRICAL TESTS - DOMAIN DESCRIPTION

In our experiments we used data from the Oncological Institute Ljubljana. They represent descriptions and diagnoses of patients in two oncological domains (Kononenko 1985). The class represents the diagnose of the patient, partially obtained by autopsy and partially in other ways. For example in the domain 'primary tumour' the location of a primary tumour in the patient is searched for.

Basically the same data were used for testing in (Clark & Niblett 1987; Michalski et al. 1986; Kononenko 1985). The only difference in our tests was a replacement of missing values by the most probable ones for a given class. Compared programs were given exactly the same data without any 'cleaning', unknown values etc. since this might be more appropriate for a specific algorithm.

The unique sequence of testing data was obtained by ten randomly chosen 70% of data for learning and remaining 30% of data for testing.

Limphography
============

number of attributes	18
no. of possible values per attribute	2 - 8 (average 3.3)
number of classes	9
total number of examples	150

distribution of examples amongst classes
number of examples in C1 to C9

1	2	3	4	5	6	7	8	9
2	1	12	8	69	53	1	4	0

importance of attributes - A1 to A18 : none of them is redundant

Primary tumour
==============

number of attributes	17
no. of possible values per attribute	2 - 3 (average 2.2)
number of classes	22
total number of examples	339

distribution of examples amongst classes
number of examples in C1 to C22

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
84	20	9	14	39	1	14	6	0	2	28	16	7	24	2	1	10	29	6	2	1	24

importance of attributes - A1 to A17 (counting how many examples overlap when omitting the i-th attribute)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
80	80	58	85	60	53	65	55	68	63	55	60	53	54	57	65	65

5. THE INFLUENCE OF DIFFERENT PARAMETERS ON CLASSIFICATION ACCURACY

The influence of impurity functions, error estimate functions and confirmation rules on classification accuracy is presented in Table 1.

LIMPHOGRAPHY				PRIMARY TUMOUR			
	INE1	INE2	GIN		INE1	INE2	GIN
AB	68.4		69.7	BA	48.3	48.4	52.3
GB	67.4		69.9	BG	48.1		51.8
II	66.8	62.6		GB	47.6		52.0
BB	66.4		70.8	IA		47.5	
IB	66.1			BI		46.8	
GI		65.9		AB	46.6		51.3
AI		65.9		BB	46.4		52.5
IG	64.8			IG	46.0		
GG	63.8			AG	45.9		
BI		63.8		AA		45.9	
IA		63.3		GI		45.7	
BG	62.6		68.4	IB	45.6		
AG	62.6			II	45.4	47.0	
BA	62.4	62.1	68.4	GG	45.2		
AA		62.4		AI		45.2	
GA		62.3		GA		45.0	

Table 1: The influence of impurity functions, error estimate functions and confirmation rules on the classification accuracy.

GINESYS (GIN) had 4 confirmation rules and preferred more general rules. GINESYS without confirmation rules is INE, INE1 preferred more specialized rules and INE2 more general ones. Initials denote the following:
I - entropy
A - percentage of majority class
G - Gini index of diversity
B - Laplacean error estimate (Niblett & Bratko 1986).
The first letter corresponds to the impurity function and the second to the error estimate.

For example IG in Table 1 denotes entropy for impurity function and Gini index of diversity for error estimate.

Remarks:
- Differences between the best and the worst combination are small regarding the fact that constructed rules differed in number and length. But note that 1% better or worse accuracy is of great importance since all reasonable systems fall into the interval (percentage of majority class, upper limit in classification accuracy).

- GINESYS with confirmation rules achieved good results no matter which combination of functions was used. It seems that the mechanism of confirmation rules is the greatest reason for significant improvements in classification accuracy. This conclusion seems plausible since confirmation rules tend to correct or at least discredit rules with the highest error rate. It also seems that the results of GINESYS are close to the upper limit of classification accuracy (Gams & Drobnic 1988).

6. DISCUSSION

Mechanisms for dealing with noise are of greatest importance in real-world domains and it is not realistic to achieve even tolerable results without them (Breiman et al. 1984; Quinlan 1986; Niblett & Bratko 1986; Bratko & Kononenko 1987; Gams 1987).

Some of the systems for empirical learning achieved good results in practical testing in several real-world domains, practically approaching or even outperforming domain experts and statistical methods (Michalski & Chilausky 1980; Bratko & Kononenko 1987). There is another relevant opinion (Breiman et al. 1984), that although all methods are more or less domain dependent, EL systems in general achieve about the same classification accuracy as other statistical methods. Since statistical methods hardly offer any explanation et all, the advantages of AI methods are obvious.

In the latest years research in machine learning provided better theoretical and practical understanding of it. Due to the new mechanisms, which have also better mathematical foundations, a new breed of empirical learning systems emerges. They take the best of both worlds: statistics and artificial intelligence, provide explanation in a human understandable way and achieve better classification accuracy than classical statistical methods. With further research another advantage of EL methods is becoming feasible - to design and implement unifying algorithms for specific families of EL algorithms.

New mechanisms such as the use of redundant knowledge seem to be of greatest importance enabling a qualitative leap ahead. It seems that the use of redundant knowledge in the proposed or further improved forms is a necessity or prerequisite for

achieving good classification accuracy. Since a rather 'ad hoc' implementation of redundant knowledge in GINESYS achieved surprisingly good results, it seems that further improvements are very probable in the near future.

GINESYS PC, a downgraded version for IBM PC, and the data used in these tests are freely available and interchangeable.

7. ACKNOWLEDGMENTS

I am grateful for suggestions from Prof. Ivan Bratko. This work was partly supported by the Slovene Research Council and COST-13 project. Research facilities were provided by the "Jozef Stefan" Institute.

7. REFERENCES

Bratko I. & Kononenko I. (1987): "Learning diagnostic rules from incomplete and noisy data", Interactions in AI and Statistics, ed. B. Phelps, London: Gower Technical Press.

Breiman L., Friedman J.H., Olshen R.A. & Stone C.J. (1984): "Classification and Regression Trees", Wadsworth International Group.

Buchanan B.G. & Feigenbaum E.A. (1978): "DENDRAL and META-DENDRAL: their applications dimensions", Artificial Intelligence, Vol. 11, pp. 5-24.

Cestnik B., Kononenko I. & Bratko I. (1987): "ASSISTANT 86: A Knowledge Elecitation Tool For Sophisticated Users", Progress in Machine Learning, (ed. Bratko I., Lavrac N.), Sigma Press.

Clark P. & Niblett P. (1987): "Induction in Noisy Domains", in Bratko I., Lavrac N. (ed.), Progress in Machine Learning, SIGMA Press.

Forsyth R. & Rada R. (1986): "Machine Learning Applications in Expert Systems and Information Retrieval", Ellis Horwood Series.

Gams M. (1987): "Unifying Principles in Automatic Learning", Ph.D. thesis, (submitted), Ljubljana.

Gams M. & Drobnic M. (1988): "Approaching the limit in classification accuracy", Informatica (submitted).

Gams M. & Lavrac N. (1987): "Review of Five Empirical Leaning Systems Within a Proposed Schemata", Progress in Machine Learning, (ed. Bratko I., Lavrac N.), Sigma Press.

Gams M. & Petkovsek M. (1988): "GINESYS: leaning from examples in the presence of noise", The 8th International Workshop on Expert Systems & Their Applications, Avignon, France.

Kodratoff Y. (1987): "Ia AI a Sub-Field of Computer Science - or is AI the Science of Explanations?", Progress in Machine Learning, (ed. Bratko I., Lavrac N.), Sigma Press.

Kononenko I. (1985): "ASSISTANT: A System for Inductive Learning", M.Sc. thesis, (in Slovene), Ljubljana.

Lavrac N., Varsek A., Gams M., Kononenko I. & Bratko I. (1986): "Automatic construction of the knowledge base for a steel classification expert system", The 6th International Workshop on

Expert Systems & Their Applications, 18-20.4.1986, Avignon, France.

Michalski R.S., Carbonell J.G. & Mitchell T.M. (ed.) (1983): "Machine Learning: an Artificial Intelligence Approach", Tioga Publishing, Palo Alto, USA.

Michalski R.S. & Chilausky L.R. (1980): "Learning by Being Told and Learning from Examples: an Experimental Comparison of Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", Policy Analysis and Information Systems, Vol. 4, No 2, pp. 125 - 160.

Michalski R.S., Mozetic I., Hong J. & Lavrac N. (1986): "The Multi-purpose Incremental Learning System AQ15 and its Testing Application to three Medical Domains", Proceedings of AAAI 86, Philadelphia, USA.

Niblett T. (1987): "Constructing Decision Trees in Noisy Domains", Progress in Machine Learning, (ed. Bratko I., Lavrac N.), Sigma Press.

Niblett T. & Bratko I. (1986): "Learning Decision Rules in Noisy Domains", Expert Systems 1986, Brighton, UK.

Quinlan J.R. (1986): "Induction of Decision Trees", Machine Learning 1, Kluwer Academic Publishers.

Watkins C.J.C.H. (1987): "Combining Cross-Validation and Search", Progress in Machine Learning, (ed. Bratko I., Lavrac N.), Sigma Press.

Waterman D.A. & Hayes-Roth F. (ed.) (1977): "Pattern-Directed Inference Systems", Academic Press.

CLASSIKA: A Knowledge Acquisition System Facilitating the Formalization of Advanced Aspects in Heuristic Classification

Ute Gappa
Universität Kaiserslautern, Fachbereich Informatik
Postfach 3049, D–6750 Kaiserslautern

Abstract

The currently most successful knowledge acquisition tools provide input formalisms that are adequate enough to enable experts entering their knowledge by themselves (for example, treatment protocols for cancer in OPAL). In respect of classification problem solving, such high level input formats have only been developed for the basic diagnostic structures (for example, rating grids in ETS and AQUINAS, failure mode hierarchy in TDE). CLASSIKA is a knowledge acquisition tool under development, the goal of which is to design knowledge acquisition techniques for advanced aspects of diagnostic problem solving. The techniques are based on the classification model of the industrially used expert system shell MED2, which offers explicite knowledge representations to handle typical classification problems. MED2's mechanisms include cost-effective data gathering, guidance of the user dialogue, plausibility check of the input data and of the final diagnoses, selection of therapy recommendations and evaluation of follow-up sessions. In particular CLASSIKA enables incremental knowledge base building while providing easy–to–use entry formalisms.

1. Introduction

Knowledge acquisition systems that enable experts to formalize their knowledge by themselves have to offer adequate and easy to understand input formalisms. A good example of such a system is OPAL [Musen et al. 87] which mimics the protocols experts use for treating cancer patients. While OPAL can exploit a very specific domain model, knowledge acquisition systems for classification such as ETS [Boose 85], AQUINAS [Boose et Bradshaw 87], MORE [Kahn et al. 85], MOLE [Eshelman et al. 87], TDE [Kahn et al. 86] can only use a general model of the problem solving type.

Classification or diagnostic problem solving – we use both expressions synonymously – can be characterized as selecting the solution from a set of possible alternatives. Many real-world problems including routine medical diagnosis and fault-finding in technical devices and processes can successfully be described as classification problems. According to the complexity of a problem the following features are required for classification shells:

— checking the input data for plausibility
— data abstraction
— efficient hypotheses generation
— cost-effective data gathering in a natural order
— rating the probability of diagnoses
— checking the final diagnoses for plausibility
— selecting recommendations

MED2 is an expert system shell for classification that provides rich mechanisms for handling these problems. Its theoretical model for classification is the starting point of the development of CLASSIKA (knowledge acquisition for classification), the concept of which is outlined in this paper. The goal of CLASSIKA is to provide knowledge acquisition techniques for multiple aspects of classification problem solving. CLASSIKA especially enables incremental knowledge base building providing easy–to–use entry formalisms.

A well known knowledge acquisition technique is the repertory grid method as realized in the Expertise Transfer System (ETS) and in AQUINAS. With the help of ETS several hundreds of knowledge bases have been built up. To a very large extent, the system facilitates experts in their work: ETS directs the dialogue with the experts and automatically generates rules out of the experts´ input. The model of the knowledge input is represented by a rating grid, i.e., a diagnosis-symptom table indicating the ratings of symptoms when a certain diagnosis is given. The diagnosis-symptom correlations are rather easy for experts to specify compared to the symptom-diagnosis correlations, i.e., the rating of how much the existence of a symptom indicates a special diagnosis. Symptom-diagnosis correlations are needed for doing classification and are derived from the table entries and from the relative importance of individual symptoms. Although the automatic generation of rules is very convenient for the expert, it involves serious problems. Since the required assumptions for statistical computation of the rules are violated in most cases (e.g., that all diagnoses are included) the rules have to be checked and modified by the expert anyway. For this task, the expert is not supported and has to switch from ETS to a different tool when extending the knowledge base. The whole problem arises from the fact that the model of the experts´ input is different from the model of the target knowledge representation (the one of classification rules). To facilitate the maintenance of the knowledge base we want the expert to determine the probability values of the classification rules directly.

In chapter 2 we describe MED2 to illustrate CLASSIKA´s underlying classification model. Chapter 3 contains a short description of the present knowledge acquisition tool of MED2 and its limitations to be overcome by CLASSIKA. CLASSIKA is outlined in chapter 4.

2. The Expert System Shell MED2

MED2 is an expert system shell for heuristic classification. Its problem solving method is based on the hypothesize-and-test stategy including differential diagnosis. MED2 also provides mechanisms for probabilistic, non-monotonic and temporal reasoning. A view of MED2´s hypothetico-deductive control strategy is given in figure 1.

The user starts a session entering his major complaints, which are selected from the set of questionsets. A questionset is a generic object type which comprises symptoms that belong

together. Completing a questionset is similar to filling out a questionaire. Within a questionset, questions are arranged in a hierarchy to achieve a natural dialogue behaviour. For example, if the user enters the symptom fever, MED2's next question would ask for the patient's temperature.
Questions can be of different types: yes/no, one-choice, multiple-choice, numerical, date (like »2 weeks ago«), duration (like »for 2 weeks«) or frequency (»2 times a week«). See figure 2 for an example.

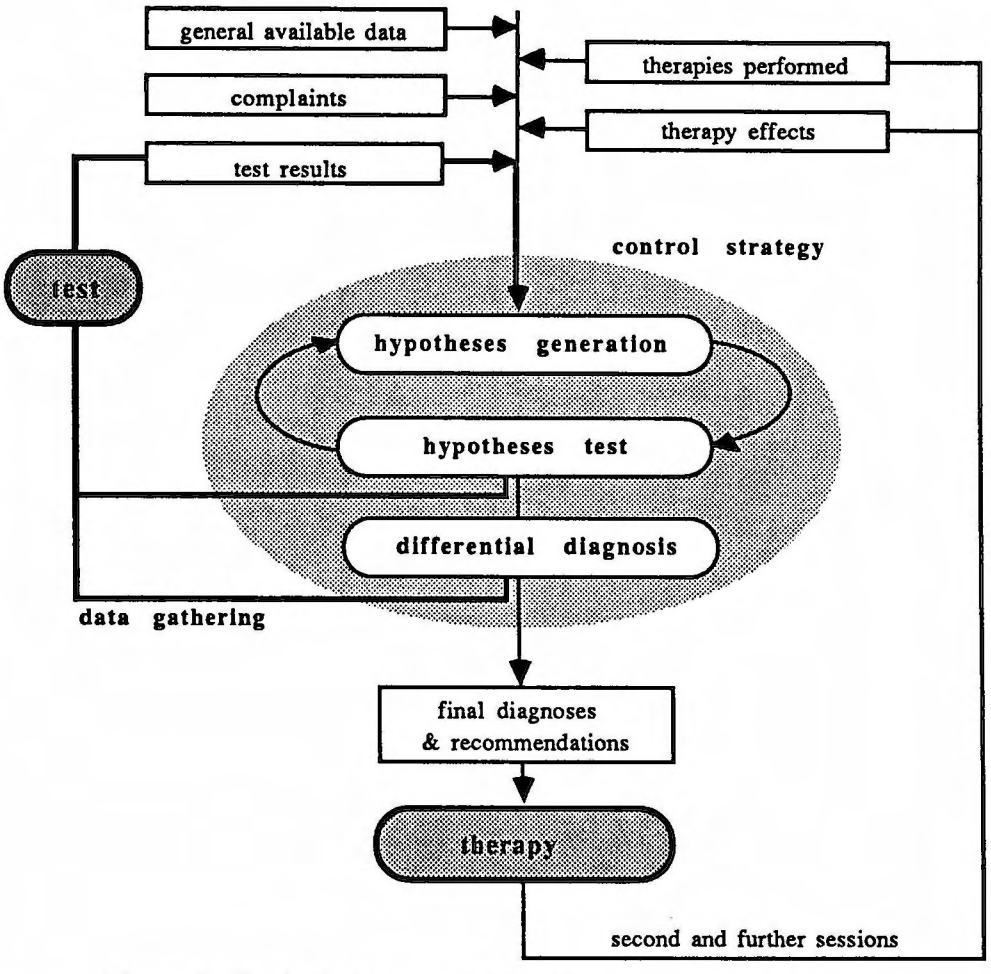


Figure 1: Control strategy of MED2

Next, MED2 checks the plausibility of the input data with consistency rules and prepares the diagnostic evaluation by aggregating the raw data to local or simple symptom interpretations. This kind of data interpretation is rather definite (categorical) compared with the probabilistic diagnostic evaluation. The simple symptom interpretations are represented the same way as questions. They can be derived by:

- *calculation of formulas* from numerical values
- *direct selection of an answer alternative*, e.g., abstracting the entered quantitative (numerical) value of fever to a qualitative one (less raised, raised, strongly raised)
- *indirect selection of an answer alternative* by combining positive and negative evidence and evaluating the resulting evidence by the help of an evaluation scheme. An example is shown in figure 9.

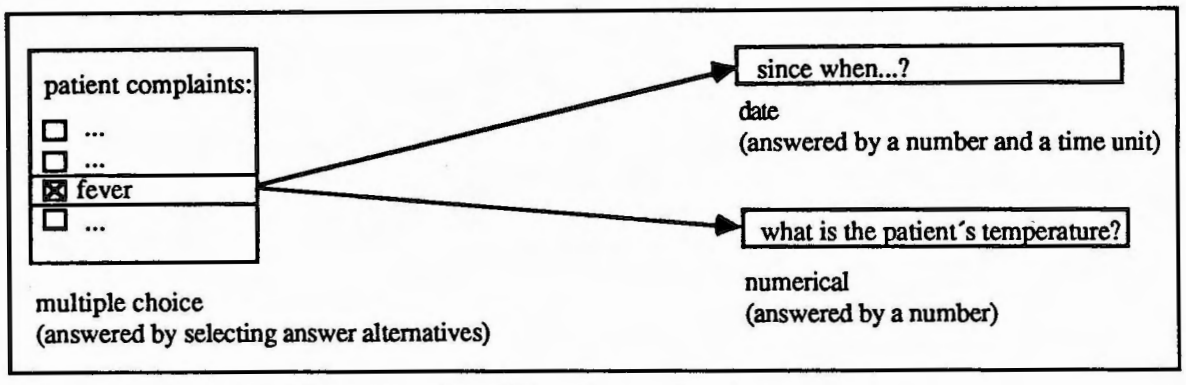


Figure 2: Question hierarchy with different question types

The following diagnostic evaluation and gathering of new data is guided by the *working memory* which contains the diagnostic hypotheses currently considered, similarly to the short term memory of humans. For the *evaluation* of the hypotheses under consideration, the currently most useful questionset is selected and presented to the user. The answers to the questions may strengthen or weaken the probability of a hypothesis (forward and backward rules) or may generate new ones (forward rules).

```
                    Explanation-mode of MED2
      evaluation of AUTOMATIC_CHOKE_SYSTEM : probable (42)
      necessary condition     :  -
      sufficient condition    :  -
      exclusion               :  -
      pro                     : probable (40)
      contra                  : neutral (0)
      value of explanation    : good (74 %)
      predisposition          : relativly seldom
      differential-diagnosis  :  -
      Variants                :      T-AUTOMATIC_CHOKE_SYSTEM

justification of AUTOMATIC_CHOKE_SYSTEM : probable (42)
justification of pro: probable (40)
P4  (20)  bec.  NORMALIZED_GAS_CONSUMPTION is TOO HIGH
activation:  STROMBERG-CARBURETOR
P4  (20)  bec.  MOTORSTART is THE ENGINE DOESN'T START UP COLD
          and   STROMBERG-CARBURETOR is established
Justification of the predisposition relativly seldom (-5)
#2 (-5)   because of the apriori-evidence of the diagnosis
Justification of the variant: T-AUTOMATIC_CHOKE_SYSTEM
P7        bec.  TYPE_OF_CAR is VW RABBIT C
          and   YEAR_OF_CONSTRUCTION inbetween  1982 and 1983

AUTOMATIC_CHOKE_SYSTEM      N
```

Figure 3: Screen layout of MED2's explanation component justifying the diagnosis "automatic choke system"

A hypothesis is evaluated by combining multiple aspects (see figure 3):

- *categorical information*: such as necessary, sufficient and exclusionary preconditions of the diagnosis
- *predisposition of the diagnosis*: The predisposition is derived from the a-priori probability of a diagnosis (its probability when not knowing any symptoms) which can be modified by rules. For example, the a-priori probability that the spark plugs of a car are broken is comparatively high, and it will even be higher if it is known that more than 15,000 kilometers have been driven since their last change.
  The predisposition by itself cannot establish or exclude a diagnosis, it can only weaken or strengthen its probability.
- *probabilistic information*: A diagnosis can be rated with six positive and six negative probabilistic categories which are analogue to the ones in INTERNIST.

Sometimes probabilistic reasoning can be replaced by specifying exceptions to a symptom-diagnosis-relation. For example, instead of saying "symptom -> diagnosis in *70%*", it may be possible to say "symptom1 -> diagnosis *unless* symptom2".

- *differential diagnosis*: Diagnoses explaining a similar set of symptoms can be specified as differential diagnoses in the sense that usually only one of them will be true, e.g., in car engine diagnosis »ignition timing too early« or »ignition timing too late«.
- *global point of view (plausibility check)*: Diagnoses that are able to explain observed, but still unexplained symptoms are more likely. Therefore their probability values are increased. For computing the explainability power of diagnoses, symptoms are grouped in characteristic units (explanationsets) rated according to their importance, and associated to potential causes (diagnoses) which are able to explain them.

The system will terminate when either all symptoms are explained by established diagnoses or all relevant data for solving the case has been gathered. On the basis of the final diagnoses, therapy recommendations (represented as diagnostic variants) are selected either by default or by rules. If a performed therapy was not wholly successful, the user can start a new session entering his performed actions and the symptoms that have changed compared to the last session. Changed symptoms and their temporal evolution, influenced by therapeutic intervention, can then give rise to new hypotheses and improved management proposals.

As this paper cannot cover all mechanisms provided by MED2, see [Puppe 87a] for more information.

3. Present Knowledge Acquisition in MED2

The present knowledge acquisition component [implemented by the author] is very closely oriented by MED2´s internal knowledge representation. To use it, the expert must already have structured and formalized his knowledge. For entering it, he instantiates generic object types which may be a questionset, a symptom (question or simple symptom interpretation), a diagnosis, a therapy variant, an explanationset or a rule. The system then asks the information relevant to this object type. This information is represented as attributes (slots) of the object. The expert´s input is checked for syntactical and, as far as possible, for semantic correctness. On request, the system explains to the expert the meaning of an attribute as well as the syntax and gives examples.

Displaying the knowledge-base or parts of it is the task of the explanation component. It also assists in testing and debugging the knowledge-base by providing justifications for the conclusions drawn by the system. In the case of an error, the knowledge acquisition component can be activated to modify the knowledge base. Without recompilations, the expert can then immediately test the effects of the modification by continuing the interrupted session.

The present knowledge acquisition tool runs on an IBM–AT in COMMON LISP and has a size of about 350 KByte including additional functions to manage the knowledge-base and to produce structured printouts. The tool has been used for building large knowledge bases in several different domains – for example, fault-finding in automatic transmissions of cars, in turbojet engines, in the production of synthetic rubber, in the painting of cars, in the production of microchips as well as the determination of mushrooms and the diagnosis of liver diseases. The

largest knowledge bases already have a size of more than 2 MByte containing more than 4000 rules.

A main shortcoming of the system is the long training period for the expert to familiarize himself with MED2´s diverse mechanisms and to structure his knowledge without help from the system. Another drawback is the long interaction time arising out of the great amount of text typing – as objects cannot be referenced by mouse selection but have to be typed in – and out of the too strong separation of explanation and knowledge acquisition component. So looking on certain aspects of the knowledge base (objects and attributes) with the help of the explanation component and directly manipulating them is not possible.

4. The Concept of CLASSIKA

Before starting to construct a knowledge base, the expert has to get familiar with MED2´s model of diagnostic reasoning, at least, in broad outlines. We intend to offer him help by commentating an exemplary system consultation. During the dialogue, a window will appear on the screen, making comments on the system´s progression and on the concepts of the underlying model and encouraging the expert to try out various explanatory functions. Comments show how to enter complaints into the system, how to revise previously entered data, why a certain questionset is activated (concept of questionset indication), why a certain diagnosis is considered as a working hypothesis (model of hypothetico-deductive reasoning), why a certain diagnosis is considered to be present (model of hypotheses evaluation), why the system proposes a certain therapeutic modality (model of therapy selection) and so on.

To avoid confusing the expert by the multitude of MED2´s knowledge representation mechanisms, CLASSIKA will ensure incremental building of the knowledge base by subsequently focusing on the various aspects of diagnostic problem solving. Initially, the main interest will be specifying findings, specifying diagnoses and relating findings to diagnoses. Additional knowledge types can be organized around this core, for example, question hierarchies, indication of questionsets, generation of hypotheses, selecting therapeutic modalities, etc. The individual techniques for acquiring different knowledge types provide special input formats, mainly forms, tables and hierarchies. These functional oriented entry formats are of a higher level of abstraction than the object oriented frames and rules of the target representation, e.g., if the expert wants to define how questionsets are indicated, he gets one form including all relevant aspects for this task; he does not have to manipulate all objects involved one by one. According to our methodology the input formalisms only differ from the target representation in the level of abstraction, but not in the kind of the underlying model. Both representations are isomorphic and can be derived from one another.

4.1. Construction of a First Knowledge Base

The core of the knowledge base consists of symptoms, diagnoses and their diagnostic relationships. CLASSIKA differs from ETS in that it includes intermediate steps of reasoning and, as already mentioned, has the expert directly rate the diagnoses. The system can adapt itself to the order of the expert´s specifications. A beginner may let the system guide him in building the knowledge base whereas someone more experienced can directly construct the

central symptom-diagnosis table (see figure 11 for an example) by specifying symptoms and diagnoses as they are needed. Following techniques are offered (figure 4):

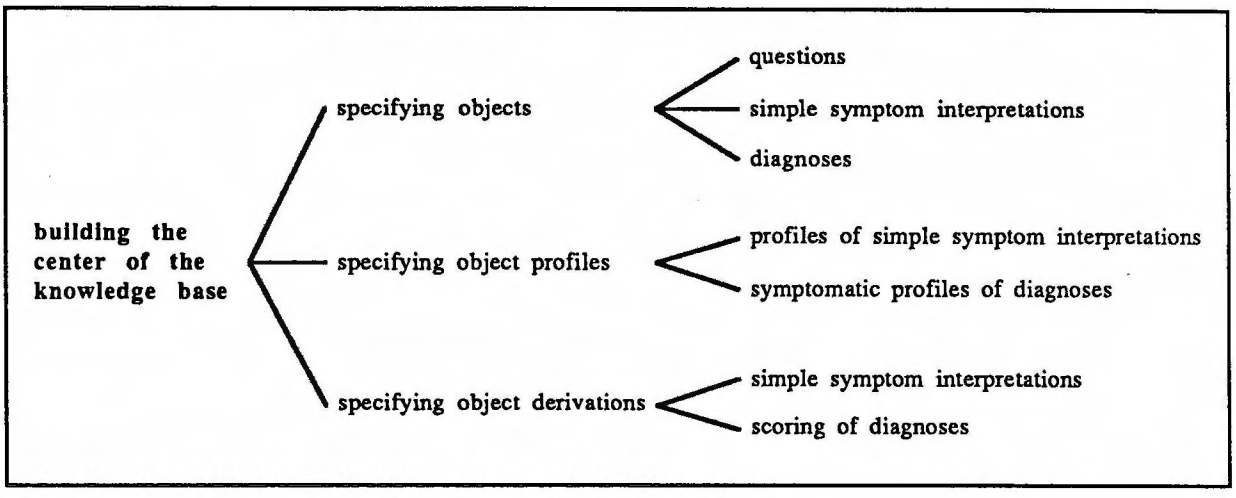


Figure 4: Techniques for building the initial knowledge base

If CLASSIKA guides the dialogue it starts asking for diagnoses (figure 5) and their symptomatic profiles (figure 6). Then the profiles are broken down to the level of questions (see figures 7, 8, 9 for an example) and finally the expert scores the diagnoses by filling out a table (figure 11). We illustrate this process by constructing a small knowledge base, which helps to decide whether an application area is suitable for building an expert system and in particular for using the classification shell MED2. For a complete description of the knowledge base see [Puppe 87a].

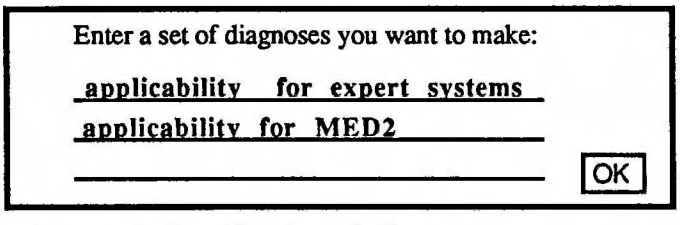


Figure 5: Specification of diagnoses

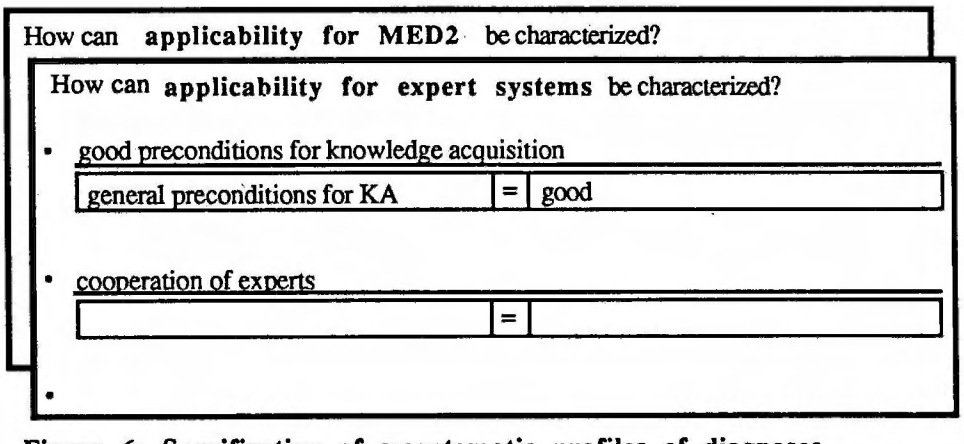


Figure 6: Specification of symptomatic profiles of diagnoses

After gathering the diagnoses to be specified, the expert defines the symptomatic profiles for each of them. The symptoms of the profiles can be of any level of abstraction. They can be refined in a next step. The expert can initially describe his criteria in "prosa" which is treated as a notebook. Relevant for the later construction of the rules is the formal description of the criteria as object–predicate–value triples (i.e., "»general precondition for knowledge acquisition« = »good«"). Entering the formal description, already specified objects and predicates (e.g.,"=") can be selected, whereas new objects need to be defined (figure 7).

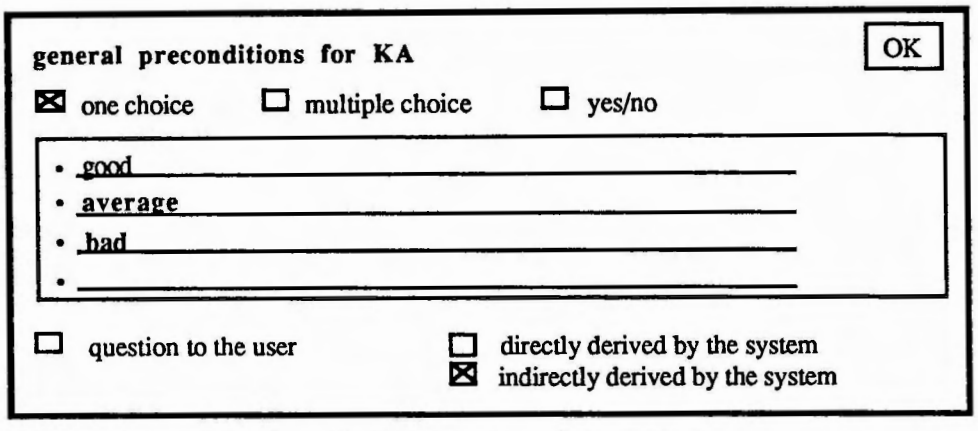


Figure 7: Specification of the symptom interpretation »general precondition for knowledge acquisition (KA)«

Symptoms which are derived from other ones can initially be defined as questions. Their derivation can be specified later. But, in our example, we want to derive the finding »general precondition for knowledge acquisition (KA)« immediately, to show the way it is to be defined. First, the criteria for its derivation are entered (»separability of the domain« etc., see figure 8). In the next step, these criteria are defined as one-choice questions with corresponding answer alternatives. Finally the indirect derivation of »general precondition for KA« is specified by filling in a generated table (figure 9). Therefore, the expert selects the necessary answer alternatives (in the example all), enters weighting factors for the symptoms or symptom combinations and defines an evaluation scheme for converting the sum of the weighting factors to a verbal description (i.e., one of its values).

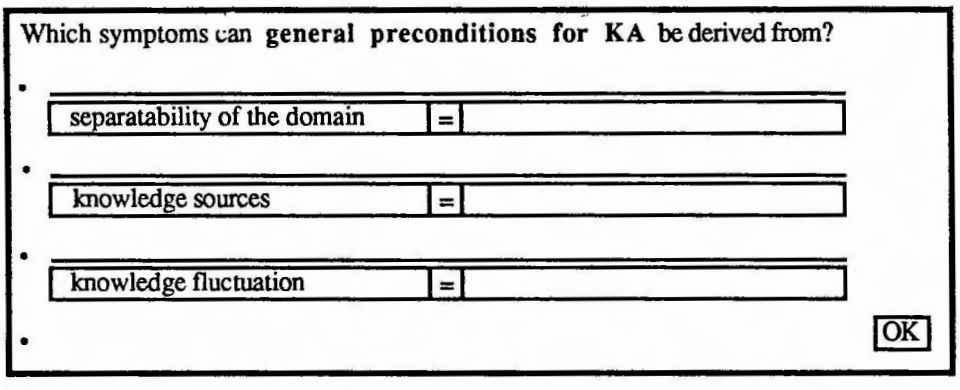


Figure 8: Specification of profiles of a symptom interpretation

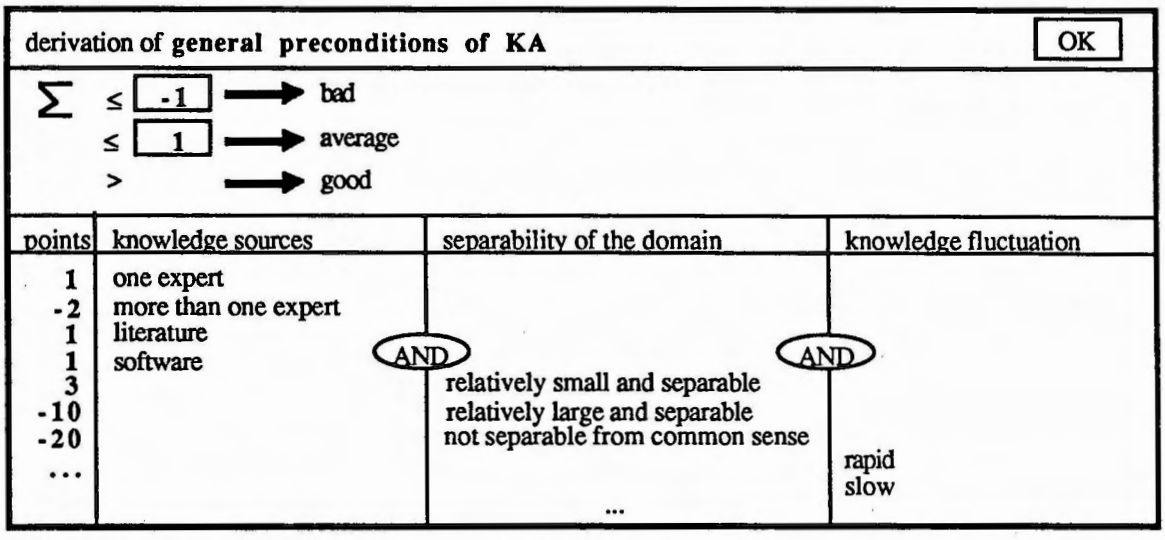


Figure 9: Indirect derivation of a simple symptom interpretation

If the expert wants to add up some other kind of knowledge as is currently focused on, he can do it. For example, he can enter additional questions (figure 10) any time.

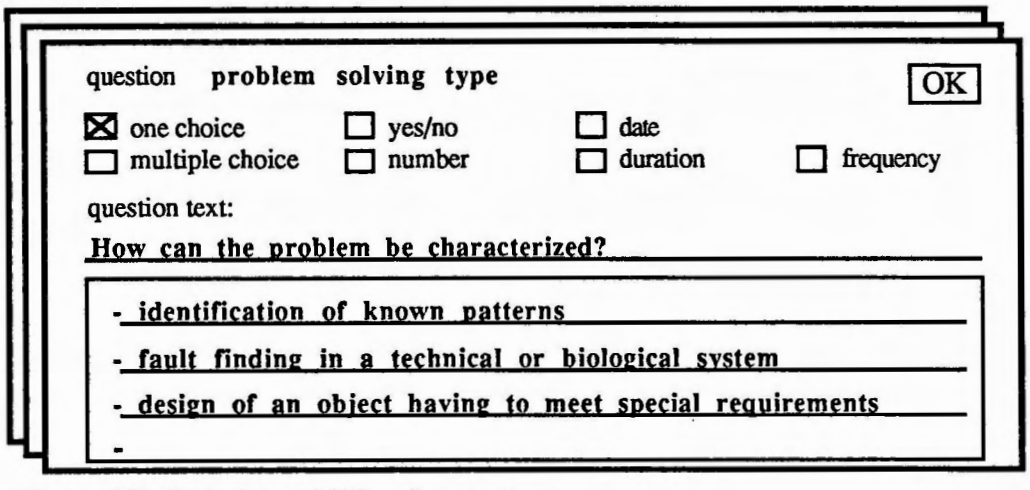


Figure 10: Entering additional symptoms

After the symptomatic profiles of diagnoses are broken down to the question level, the diagnostic relevance of symptoms for different diagnoses is stated, i.e., how strong a symptom´s existence or non-existence argues in favor of or against a diagnosis. For this, the expert selects diagnoses which he wants to consider. Then the system generates a table with the symptoms included in the diagnoses´ symptomatic profiles. For every symptom-diagnosis relation, two boxes appear. The first refers to the existence of the symptom and the second to its non-existence (see figure 11). Since non-existence of a one-choice symptom is often itself named (e.g., »general preconditions of knowledge acquisition« = »bad«, instead of NOT »good«), the second rating of one-choice symptoms is not always needed. By offering two boxes for each relation, the expert is remembered to assess the valuable information that a certain symptom is *not* given .

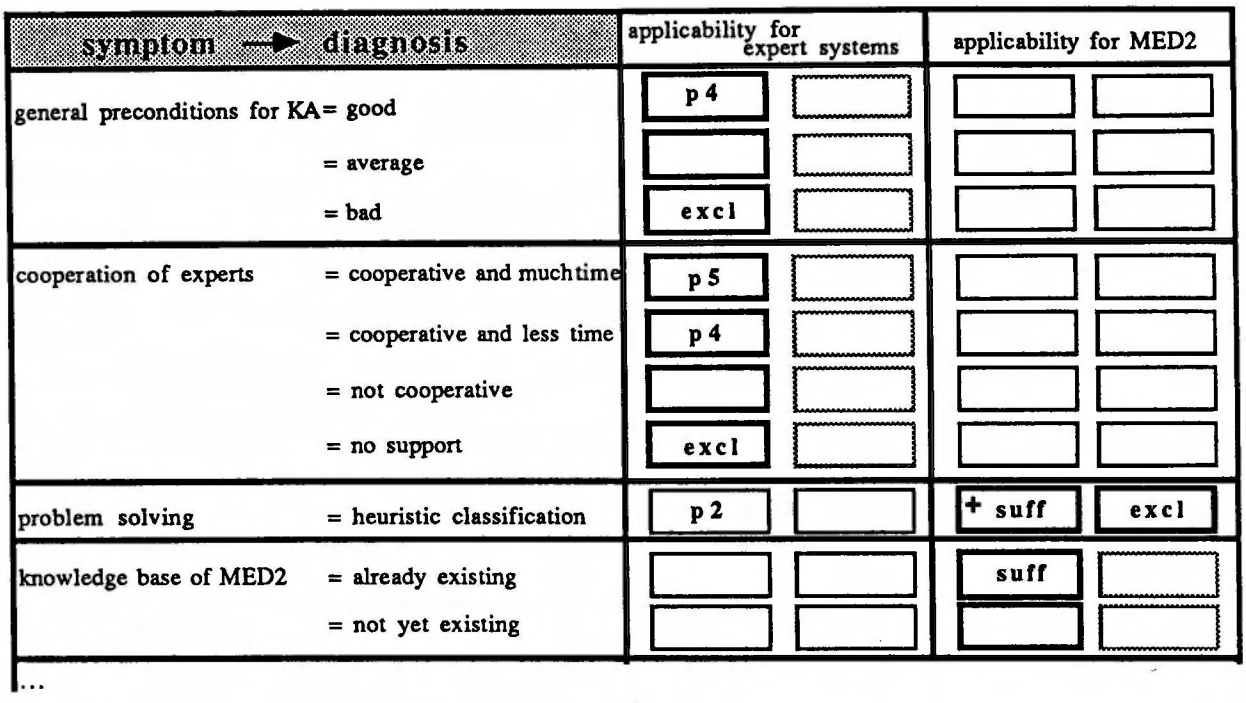


Figure 11: Table for scoring of diagnoses

Every table entry exactly corresponds to a production rule in MED2. On the left hand side of the table, the core condition of the rule is presented. It refers to exactly one symptom. If the condition is more complex (e.g., combining several symptoms), the expert can enter this knowledge in a special rule entry format (see figure 12), which appears on the screen, when he "opens" the respective box (by clicking into it). The existence of a more complex condition is indicated by a "+" in the boxes of the table.

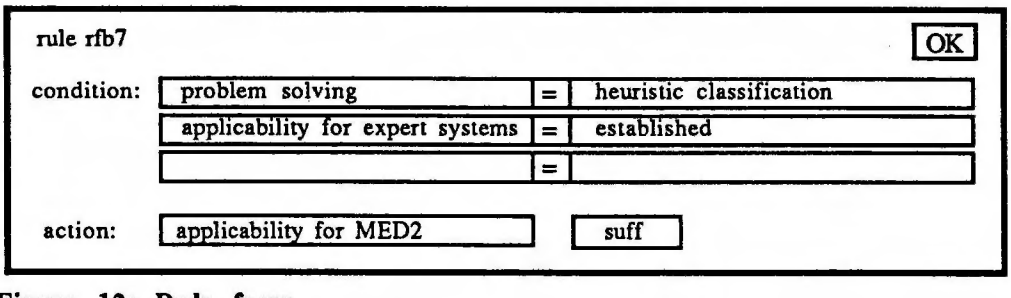


Figure 12: Rule form

Besides necessary, sufficient and exclusionary conditions (abbreviated by necc, suff and excl), probabilistic score categories can be used in order to evaluate diagnoses. Probabilistic reasoning will be needed to express uncertainties in making a diagnosis. Therefore, positive and negative score categories (p1, ... , p6, n1, ... , n6) are used. They are combined so that two categories of one level correspond to one of the next level (for example, p5 = p4 + p4). If no categorical information contradicts it, a diagnosis is confirmed when its score is greater than p5. The probabilistic score can be modified by the predisposition of a diagnosis, which can be explicitly represented. However, it isnot used in the example.

The knowledge base thus constructed is ready to be run and tested.

4.2. Incremental Extension of the Knowledge Base

The expert can extend his knowledge base by giving new symptoms and diagnoses, by incorporating them into the table, and by configuring new tables for whatever correlations he wants to consider. Preconditions of rules can also refer to combinations of symptoms (AND-connected) of which only the main one appears in the table – the first one is interpreted as the main one as a convention (see figure 11 and 12). Furthermore, for every rule, contexts and exceptions can be given. A complete rule form is shown in figure 15.

The methods described so far are sufficient to acquire and represent just the basic structure of diagnostic problem solving. For modeling the more complex diagnostic requirements mentioned in the introduction, additional knowledge acquisition techniques are needed. Proceeding from the "basic structure" of the knowledge base, CLASSIKA offers a lot of techniques for entering additional kinds of knowledge concerning the various aspects of diagnostic reasoning (figure 13). The individual methods are are based on the diagnostic model of MED2 (e.g., concept of questionsets) and are especially designed to support incremental knowledge base development. The techniques can be activated in an arbitrary order.

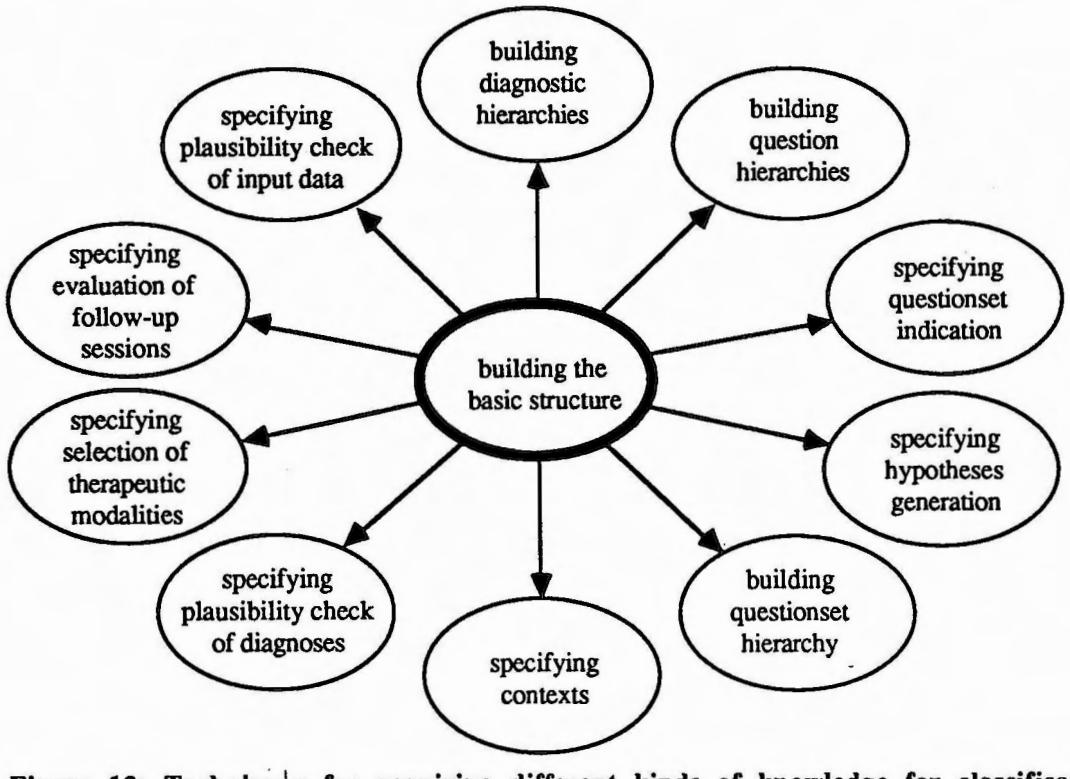


Figure 13: Techniques for acquiring different kinds of knowledge for classification

In the following, we describe the expert-directed modes of the techniques. The system-directed mode arises, if the system asks and fills in the knowledge which, in the currently considered part of the knowledge base, can explicitly be represented. The system-directed mode is very useful for beginners, but it is unavoidable that a lot of unnecessary questions will be asked. As long as the system lacks additional knowledge, it must systematically ask for all kinds that can be represented. For example, when filling in a symptom-diagnosis table, all possible entries have to be asked for (as in the repertory grid techniques) while only the least are relevant. When the expert knows how to enter the individual kinds of knowledge, he will prefer taking the initiative of the knowledge base building because it is much more efficient. Besides, experien-

ces with the mixed-initiative knowledge acquisition tool TDE showed that the system-directed mode has mainly been used by beginners.

The individual techniques for acquiring multiple kinds of knowledge for classification are described in the following.

- **Building diagnosis hierarchies**
  When this technique is activated, CLASSIKA displays the structure of the diagnoses already implicitly specified by table entries in the form of diagnosis-diagnosis rules. By extending the hierarchies (eventually multiple hierarchies or networks) the expert represents the step-by-step proceeding of the problem solving. Each arc of the hierarchy corresponds to a rule. If a new arc is drawn, the created rule has to be completed by determining its strength and eventually by placing further symptoms in the rule´s condition.

- **Building question hierarchies**
  For building question hierarchies, the expert specifies conditions under which a certain question should be asked. The hierarchy can be graphically constructed, the conditions being specified in the same formats as used for entering rule conditions. Once a hierarchy of questions is established, not all questions need to be asked when running the knowledge base.

- **Specifying questionset indication**
  If not all question hierarchies (e.g., Was test X performed?) should be routinely asked, they are summarized as questionsets. These can be activated on demand by the system or by the user. In knowledge acquisition, the expert has to define the system-directed indication of questionsets. For this, the following table is offered to him (figure 14).

questionset indication	»questionset a«	»questionset b«	»questionset c«	...	
direct indication					
»condition k«		ind			ind or asknot
»condition l«	asknot				
...					
costs					
apriori-costs		-5			negative numbers
»condition i«					
...					
benefits					
»diagnosis m«		10			
»diagnosis n«	20				positive numbers
»diagnosis o«	...				
...			...		

Figure 14: Table for questionset indication

Questionsets can be categorically indicated (entry: ind), for example, "if a patient complaints about chest pain, do an ECG" or categorically contra-indicated (entry: asknot) "if heart infarction cannot be ruled out, do not do an treadmill examination". If there are still hypotheses that need to be evaluated after the user selected and the categorically indicated questionsets are asked, MED2 will indicate that questionset which is most useful for testing the actual hypotheses and which costs are the lowest (cost/benefit analysis). The static costs of a questionset (e.g., risk, delay, expenditure) can be modified by rules, e.g., to represent the risk of a patient to undergo a special test. Except for the a-priori costs which are represented as an attribute of the questionsets, each table entry corresponds to a rule.

- **Specifying hypotheses generation**
  In order to avoid searching the complete diagnostic space, hypotheses have to be generated selectively. This means that the default entries "hypothesis generation" of those rules, which should be used only for evaluation of an already hypothesized diagnosis, have to be deleted (figure 15).

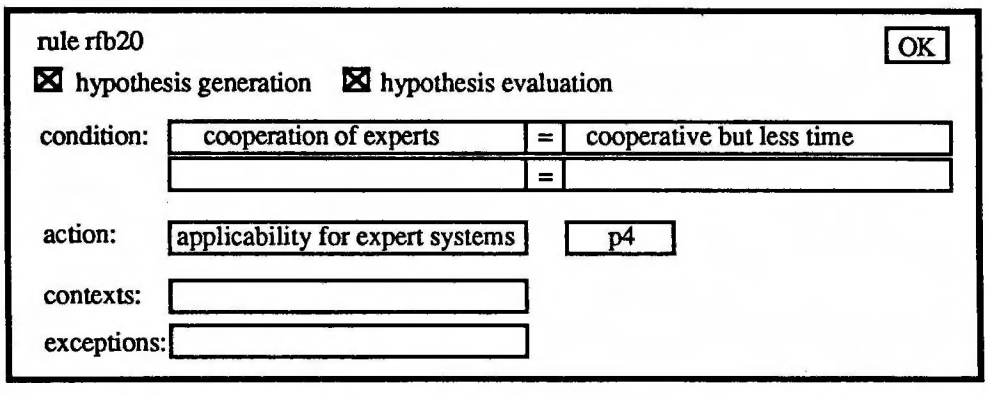


Figure 15: Complete rule form

Another way to delete the entries is by generating symptom-diagnosis tables which only indicate rules for hypothesis generation.

- **Building questionset hierarchy**
  On the basis of the existing questionsets new questionsets are created and put on top of the existing ones. The new questionsets do not contain questions. Their purpose is to facilitate the selection of a questionset. It helps the user, e.g., to enter his complaints at the beginning of a session.

- **Specifying contexts**
  The activation of a rule can be made conditional on a certain context. For example, in engine diagnosis, for each different engine type an own context can be established, so that the special rules need only be activated in their context (internally a context is represented as a special type of a diagnosis).

- **Specifying plausibility check of the diagnoses**
  To check the final diagnoses for plausibility, explanationsets can be specified which condense individual symptoms to units that need to be explained by diagnoses. For every explanationset, diagnoses must be given which are able to explain it. Special tables are offered to enter the knowledge relevant to derive explanationsets out of individual symptoms.

- **Specifying selection of therapeutic modalities**
  For therapy selection, diagnostic variants are instantiated which represent different therapeutic modalities (figure 16). The expert may specify a questionset which will ask for special information relevant for the therapy selection. This questionset will be asked as soon as the diagnosis is established. The selection of therapies can be specified in a special table using probabilistic ratings (figure 17). A therapy variant can also be selected by default if no further information is available or needed.

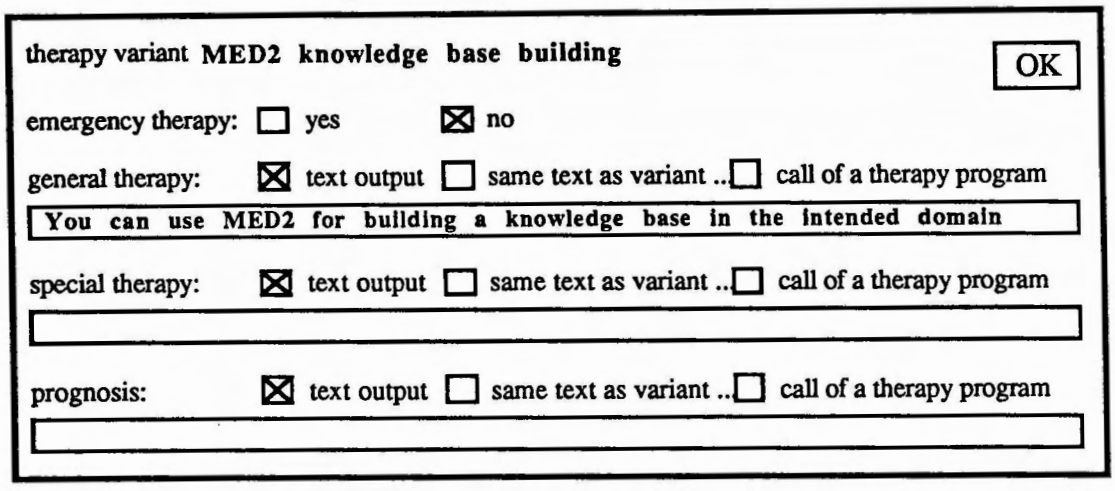


Figure 16: Specification of a therapy variant

selection of therapy variants	»therapyvar r«	»therapyvar s«	»therapyvar t«	»therapyvar u«
diagnoses				
»diagnosis i«	possible	possible	default	
»diagnosis j«			possible	
...				
conditions				
»condition k«	p4			
»condition l«		p2	...	
...	...			...

Figure 17: Table for selection of therapy variants

- **Specifying evaluation of follow-up sessions**
  In a follow-up session, the user will first enter the time passed since the last session. Then he will tell the system which therapy or repair actions he has performed and how the symptoms have changed. For knowledge acquisition this means that the expert has to define socalled therapy questionsets which explore the details of the performed therapies. These questionsets can then be selected by the user, when he wants to enter his performed actions. The second aspect is the evaluation of "histories" of changing symptoms. For specifying this, the expert may use special predicates which are offered in special forms for entering time-dependent conditions. With them, he is supported by formulating rule conditions like "if the degree of fever is fallen by at least one degree within one day after taking a special medicine then ..."

- **Specifying plausibility check of input data**
  For numerical or time related questions possible ranges can be defined (e.g., age: 0 – 120), so that only answers within the range will be accepted. To detect inconsistencies of different answers, the expert may characterize inconsistent situations by rules. For this, forms for entering rule conditions are filled in. If such an inconsistency is detected during run time, MED2 will call upon the user to correct one of the regarding inputs. Finally, if it is possible to recognize automatically transfered data from a measuring instrument as being false, a diagnosis which says "instrument X is broken" may be established and set as an exception to the rules that directly evaluate the data. This effects that the data is not processed.

4.3. Knowledge Editor with Full Functionality

Not all kinds of knowledge which can explicitly be represented in MED2 are covered by the techniques discussed. For entering additional knowledge as well as for reviewing the knowledge base, CLASSIKA falls back to the underlying object-centered editor. The objects of the knowledge base can be traversed with a browser working on the various hierarchies. For the individual objects, the frames can be inspected which include attributes of the object as well as relationships to other objects of the knowledge base. The frames for rules and therapy variants have already been shown (figure 15 and 16). Figure 18 illustrates the attributes of a diagnosis frame.

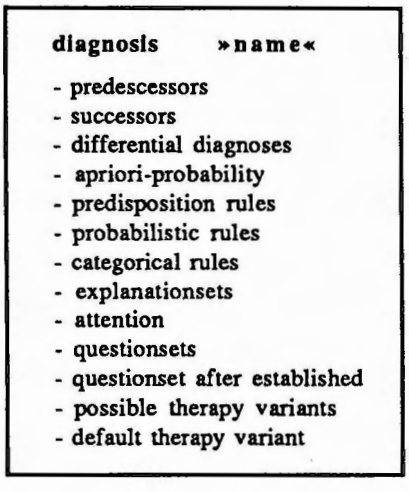


Figure 18: Frame attributes of a diagnosis

5. Discussion

The knowledge representation of an expert system plays a crucial role for knowledge acquisition. MED2´s highly structured representation provides a very strong and adequate model for meeting typical classification requirements. The kinds of knowledge which are explicitly represented refer to cost-effective data-gathering, generation of hypotheses and their evaluation by combining multiple criteria such as categorical conditions, probabilistic evidence, predisposition, differential diagnoses and explanation power of a diagnosis. Furthermore mechanisms include checking the input data, proposing recommendations and evaluating follow-up sessions. Exploiting MED2´s model of classification, CLASSIKA can offer multiple knowledge acquisition techniques supported by easy-to-understand input formats with a high level of abstraction.

CLASSIKA differs from a lot of other knowledge acquisition systems for classification in the richness of the kinds of knowledge which are acquired. Most other systems have a much simpler knowledge representation covering only the core of diagnostic reasoning. Taking AQUINAS, for example, the acquired knowledge roughly corresponds to CLASSIKA´s "basic structure". A main difference to TDE (resp. TEST [Kahn 87a]) is, as far as we can see, it does not support non-monotonic and probabilistic reasoning. MED2 is especially designed for reasoning with uncertainty in respect to hypotheses evaluation as well as selection of diagnostic tests and therapies and handling exceptions. For example, in the domain of thoracic pain diagnosis [Puppe 84], only few symptoms are existing which allow categorical reasoning.

Another main difference is that CLASSIKA gives the expert the initiative for knowledge base building while AQUINAS, MORE and its successor MOLE want the system to have the initia-

tive. The latter use static and dynamic analysis methods for determining which knowledge to ask for. In a complex domain using a rich knowledge representation, it is not feasable to have the system ask for the information because there are too many possibilities for knowledge base extension as well as error correcting. Therefore, we want to support the expert in building the knowledge base by his own initiative. This requires that the theoretical model of the system is transparent to the expert and that the input formalisms are easy to use and of an adequate level of abstraction. Additionally, powerful explanation facilities are provided so that the expert can trace the problem solving process and locate incorrect or missing knowledge.

If the expert directs knowledge base building, knowledge transfer will be more effective, which is the main goal of knowledge acquisition tools.

Acknowledgements

I want to thank Frank Puppe for his contribution to the paper's ideas and his review. I am also grateful to Horst Fischer and Ann Smith for their comments on earlier drafts of the paper.

References

[Boose 85] John H. Boose
A knowledge acquisition program for expert systems based on personal construct psychology
International Journal of Man-Machine Studies, Vol. 23, 1985

[Boose et Bradshaw 87] John H.Boose, Jeffrey M. Bradshaw
Expertise transfer and complex domains: using AQUINAS as a knowledge acquisition workbench for knowledge-based systems
International Journal of Man-Machine Studies, Vol. 26, 1987

[Eshelman et al. 87] Larry L Eshelman, Damien D. Ehret, John McDermott, Ming Tan
MOLE: a tenacious knowledge acquisition tool
International Journal of Man-Machine Studies, Vol. 26, 1987

[Kahn et al. 85] Gary Kahn, Steve Nowlan, John McDermott
MORE: an intelligent knowledge acquisition tool
Proceedings of the IJCAI, 1985

[Kahn et al. 86] Gary S. Kahn, Edwin H. Breaux, Peter DeKlerk, Robert L. Joseph
A mixed-initiative workbench for knowledge acquisition
Proceedings of Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada, 1986

[Kahn 87a] Gary S. Kahn
TEST: a model-driven application shell
Proceedings of the AAAI, 1987

[Kahn 87b] Gary S. Kahn
From application shell to knowledge acquisition system
Proceedings of the IJCAI, 1987

[Musen et al. 87] Mark M. Musen, Lawrence L. M. Fagan, David M. Combs, Edward H. Shortliffe
Use of a domain model to drive an interactive knowledge-editing tool
International Journal of Man-Machine Studies, Vol. 26, 1987

[Puppe 87a] Frank Puppe
Diagnostic problem solving with expert systems (in German)
Informatik Fachberichte 148, Springer, 1987

[Puppe 87b] Frank Puppe
Requirements for a classification expert system shell and their realization in MED2
Applied Artificial Intelligence 1, 1987

[Puppe 84] Bernhard Puppe
The history of computer-assisted medical decision making and MED1, an expert system for thoracic pain diagnosis (German)
Dissertation, Universität Freiburg, 1984

INC^2 : A Prototype-Based Incremental Conceptual Clustering System

Mirsad Hadzikadic

Department of Computer Science
The University of North Carolina at Charlotte
Charlotte, NC 28223
USA

Abstract

Learning by observation is an important area of machine learning concentrated on the process of creating a hierarchy of concept descriptions over a set of given events, thus summarizing experience. In this paper the incremental conceptual clustering system, INC^2 is presented. Input to INC^2 is a set of events, given to the system one at a time. INC^2 is to recognize the patterns of similarity among events and describe them in a form of a hierarchy of generalized non-disjoint concept descriptions. It carries out a hill-climbing search through a space of hierarchical classification schemes. It uses operators that let the system search in both directions of this space. Events are then stored under the generalization(s) that describe them. The resulting concept hierarchy can be used by a performance system to make inferences about new, previously unseen, instances.

The main topic: machine learning.

1. Introduction

Classification is one of the basic mechanisms which provides humans with an ability to deal efficiently with the complexity of the real world. This process is a powerful data reduction tool used by cognitive systems to help in solving everyday problems. Fisher and Langley (1985) have explained classification processes as being composed of three distinct but inter-dependent subprocesses: the process of deriving a hierarchical classification scheme, the process of aggregating events into individual classes (clustering), and the process of assigning conceptual descriptions to event classes (characterization).

The characterization component is shared by the classification task and learning from examples. Since there has been a substantial amount of work done on systems that learn from examples (Quinlan, 1983, 1986; Winston, 1986; Vere, 1975), the natural approach to building a classification system would be to solve the clustering problem first, and then to solve the characterization problem by employing one of the traditional methods of learning from examples. In fact, present conceptual clustering algorithms (the most popular method of classification) can be characterized in this way (CLUSTER/2, CLUSTER/S - Michalski and Stepp, 1983, 1984, 1986; GLAUBER - Langley *et al.*, 1986; IPP - Lebowitz, 1983; RUMMAGE - Fisher, 1984; DISCON - Langley and Sage, 1984). A thorough analysis of these algorithms is given by Fisher and Langley (1985).

A system described in this paper, INC^2 (INcremental Conceptual Clustering system), learns from observation by recognizing regularities among events (examples, objects, etc.) and organizing them into a hierarchy. Lebowitz (1987) defined *learning from observation* as a task that is important in domains where events are not pre-classified, but where one still

wishes to detect general rules and intelligently organize events. INC² takes a series of events (described as collections of attribute-value-relevance triplets) and builds up a hierarchy of concepts which captures similarity patterns among the events. A performance element then uses such concept descriptions to make inferences about new events based on partial information.

The operation of INC² is characterized by the following features (emphasized by Lebowitz, 1987):

- It learns by observation;
- It is incremental;
- It handles events in large numbers (in fact, the larger the number of events the better the hierarchy of concepts);
- Its generalizations are pragmatic (they do not perfectly describe all the instances they cover).

Although INC² does not represent a psychological model, it demonstrates some properties usually attributed to humans, such as a prototypical representation of concepts, 'stronger link' between a concept and its more prototypical instances, etc. The concept representation mechanism implemented in INC² is based on the research done by Rosch, Mervis *et al.* (Rosch, 1973; Rosch and Mervis, 1975; Rosch *et al.*, 1976; Rosch, 1978), who have hypothesized that the members of categories which are considered most *prototypical* are those with the most attributes in common with other members of the category and the least attributes in common with other categories.

Basically, the task of INC² is that of conceptual clustering as defined by Michalski and Stepp (1983) and Fisher and Langley (1985). However, this

work was mainly influenced by the similar incremental conceptual clustering systems reported in the literature, such as COBWEB (Fisher, 1987), UNIMEM (Lebowitz, 1987), CYRUS (Kolodner, 1984), and the system dscribed by Hadzikadic (1987).

2. Knowledge Representation

The evidence that *prototypes* play a critical role in human categorization is compelling. Rosch has demonstrated (1973, 1978) the existence of prototypes for both natural categories (like colors and animals) and artificial categories (like dot patterns and schematic drawings). In this work, a *schema* mechanism is adopted to implement a concept of prototype. A schema is a *declarative* structure that organizes pieces of knowledge related to the same entity into a unitary whole. In various implementations, schemata correspond to Minsky's *frames* (1975) and Schank and Abelson's *scripts* (1977).

A template of the schema structure used in this work, representing a concept *schema* $_i$, is given below:

name: schema$_i$
attribute: (attribute$_k$, value$_k$, relevance$_k$)
specialization-of: (superordinate-concept$_l$, strength$_l$)
generalization-of: (subordinate-concept$_m$, strength$_m$)
instance-of: (concept$_n$, strength$_n$)
has-instance: (instance$_q$, strength$_q$)

The *specialization-of*, *generalization-of*, *instance-of*, and *has-instance* slots contain the strength of the class-subclass and class-instance links, respectively.

3. System Description

Input to INC² is a series of events (examples, instances), given to the system one at a time. The goal is to recognize patterns of similarity among

events (if they exist) and describe them in a form of a hierarchy of generalized non-disjoint concept descriptions. INC² carries out a hill-climbing search through a space of hierarchical classification schemes. It uses operators that let the system search in both directions of this space. Events are then stored under the generalization(s) that describe them. The resulting concept hierarchy can be used by a performance system to make inferences about new, previously unseen, instances.

3.1. Generating a Prototypical Description

All (attribute, value) pairs used in the descriptions of given events should be represented in the class description, once the class membership is determined. The more (attribute, value) pairs in the class description, the more specific the description, for an event must match more attributes and their values to claim membership. The relevancy *rel* of a (attribute, value) pair (att, val) in the description D of the class c is computed as a sum of relevances of the class members with that particular property, divided by the total number of members, i.e.:

$$rel(c, att, val) = \frac{\sum_{i=1}^{m} rel(e_i, att, val)}{n} \qquad (1)$$

where $e_i \in c$, $(att, val, rel) \in D_a(e_i)$, $D(e_i)$ is a description of the event e_i, $D_a(e_i)$ represents a subset of $D(e_i)$ containing the (attribute, value) pairs only (no hierarchical links), n is the cardinality of the class c, and $m \leqslant n$ stands for the number of events $e_i \in c$ with the property (att, val). In order to keep the whole process efficient, the procedure for computing the relevance of a (attribute, value) pair is implemented in INC² in a slightly different way in. The system keeps track of all the instances stored under any class, thus being able to reconstruct the summary information about previously seen

instances very easily, without having to look at them over and over again.

This is an effective way of generating a prototypical description of a class. Also, to improve the efficiency of the overall process as well as the subsequent retrieval, the (attribute, value) pairs with the relevance below the prespecified threshold (0.2 in our case) should be dropped at this point from the concept description, since they do not contribute significantly to the prototypicality effect.

3.2. Link: Degree of Class Membership

Once the class description has been generated (after seeing a first event this will be just a singleton class) and the corresponding relevances calculated, the system can proceed and receive another event. When trying to determine the degree of match between a set of existing concepts and an event, in attempt to decide whether to create a new class or place the event under one of the existing generalizations, INC² computes the degree of class membership with all top-level classes for the given event. In order to do that, the system will sum up the products of relevances for the attributes that have the same value in the descriptions of both the class and event. (Notice that the relevance of all attribute-value pairs in the event's description is equal to 1.0 - it simply posses that particular value for that particular feature). The resulting sum is then divided by the number of (attribute, value) pairs (no links) in the class description. This process is summarized in the following equation:

$$link(c, cm_j) = \frac{\sum_{i=1}^{n} s(c, att_i, val) \times s(cm_j, att_i, val)}{n} \qquad (2)$$

where $link(c, cm)$ stands for the strength of the link between the class c and its potential member cm_j, i.e. $cm \in c$, $s(c, att_i, val)$ represents the relevance of

the (att_i, val) pair in $D_a(c)$, $(att_i, val, s) \in D_a(c)$ and $(att_i, val, s) \in D_a(cm)$, and n is a cardinality of $D_a(c)$.

To determine to which (if any) of the existing top-level classes c_i the new event e belongs to, the system should compute $link(c_i, e)$ for $i = 1, \cdots, n$ where n stands for the number of classes under consideration. The event, then, belongs to the class which both maximizes the value of *link* and is greater than the prespecified threshold (0.5 in our case). The corresponding class description is subsequently updated according to expression 1. The event is, then, compared to the subclasses of the 'best' class in order to find both best and most specific host class. This procedure is, then, invoked recursively. If, on the other hand, there is no best host for the event, than INC² creates a new singleton class with that event as the only instance. Expression 2 is used to calculate the strength of the hierarchical links between classes and subclasses (instances).

3.3. Operators

Similarly to COBWEB (Fisher, 1987), there are four operators used by INC² to generate a hierarchy of concepts:

- *Place* - placing an event in an existing class. Once the best host is determined, simply calculate the new relevances of the (attribute, value) pairs in the class description according to expression 1.
- *Create* - creating a new class. A description of the new class is exactly the same as a description of the event.
- *Merge* - merging two or more classes into one. This operator is applied when there is more than one 'best' host for an event. A new class is introduced which replaces all of the best hosts. The participating classes (best hosts) are stored as subclasses of the newly created class.

- *Split* - splitting a class into two or more classes. When there is no good host for a given event, INC² does not create automatically a new class. It first takes the best out of no good hosts and evaluate its subclasses. If it finds a good host(s) among them, then it replaces the best of no-good-hosts with its subclasses, and the process continues.

3.4. Algorithm

The control strategy used by INC² is summarized below

Procedure INC² (Event, Root)

1) Evaluate all the children of Root.
2) If one best host is found, then update its description and call INC² (Event, BestHost).
3) Otherwise, if more than one best host is found, then apply the operator *merge*, update the description of the newly generated class, and for each best hosts update its description and call INC² (Event, BestHost).
3) Otherwise, if no good host is found, then evaluate the children of the best of no-good nodes and try to find a best host.
   a) If found, then apply the operator *split*, update a description of the best host, and call INC² (Event, BestHost).
   b) Otherwise, create a new singleton class.

A simple example of the result produced by this algorithm is given in table 2. The actual input is given in table 1 (taken from Fisher, 1987). As an example, a description of one class ('amphibian'/'fish') is given in table 3. For the sake of simplicity, the hierarchical links are not included.

Table 1.

Name	BodyCover	HeartChamber	BodyTemp	Fertilization
'mammal'	hair	four	regulated	internal
'bird'	feathers	four	regulated	internal
'reptile'	cornified-skin	imperfect-four	unregulated	internal
'amphibian'	moist-skin	three	unregulated	external
'fish'	scales	two	unregulated	external

Table 2

'animals'				
'mamal/bird'		'reptile'	'amphibian/fish'	
'mamal'	'bird'		'fish'	'amphibian'

Table 3

'amphibian/fish'		
attribute	value	relevance
BodyCover	moist-skin	0.5
BodyCover	scales	0.5
HeartChamber	3	0.5
HeartChamber	2	0.5
BodyTemp	unregulated	1.0
Fertilization	external	1.0

4. Analysis

The system has been applied to the domain of liver-related diseases. The input data consisted of the results of 8 tests, with yes/no type of answers. Not all of the tests were administered in all cases, and the input matrix was very sparse. The data contained noisy data as well (the same event assigned to more than one class by an expert, as well as incomplete data). INC² was to detect 8 different classes of diseases, as suggested by an expert. The expert's performance in this particular case was no better than 70% (on new unseen examples). INC²'s performance depended on the number of training instances. It took more than a hundred training instances to come up with the performance that was comparable to the expert's performance. The limited space does not allow us to go into details of this experiment, but some general remarks should be given about the system in general:

- INC² learns by observation.
- It is incremental.
- It uses a hill-climbing through the space of hierarchies, although the *merge* and *split* operators help the system to correct some of the 'mistakes' (trough some form of backtracking).
- It produces non-disjoint classes.
- It is based on the prototypical representation of concepts.
- It produces a classification which can be used by the performance element to make inferences about new events from the outside world.
- It is robust with respect to noisy data because of the way the relevance of the (attribute, value) pairs in a concept description is computed, as well as the fact that the pairs with the relevance below the prespecified

threshold are dropped (over time) from the concept's descriptions.

- It needs a lot of examples before it can produce a stable hierarchy of concept descriptions.
- It is sensitive to the values of the thresholds introduced in the paper, although this effect decreases with the number of seen events.

UNIMEM, CYRUS, and COBWEB systems are very similar to INC². The difference is in the nature of operators and representational formalisms. Of course, these differences have as a consequence a different search control mechanism. It would be useful to compare these systems on a similar set of problem domains, although Fisher (1987) has provided a comparison of UNIMEM, CYRUS, and COBWEB systems along several dimensions.

5. Conclusion

Incremental concept formation is an important area of machine learning. Its success would ease the problem of knowledge acquisition in knowledge-based systems. In this paper an incremental conceptual clustering system, INC², is described, which generates a hierarchy of concept descriptions over a set of input events, given one at a time. Future work will focus on removing some of INC²'s limitations, such as

- limited events and concepts description formalism,
- search strategy,
- nature and number of available operators,
- static nature of the thresholds being introduced.
- system's sensitivity on the threshold values.

References

Fisher, D. (1984). A Hierarchical Conceptual Clustering Algorithm. *Technical Report*, Department of Information and Computer Science, University of California, Irvine.

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

Fisher, D. (1987). Knowledge Acquisition Via Incremental Conceptual Clustering. In *Machine Learning*, **2**, 2, 139-172.

Hadzikadic, M. (1987). Concept Formation by Heuristic Classification. *Unpublished Doctoral Dissertation.* Southern Methodist University, Dallas.

Kolodner, J. L. (1984). Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model. Lawrence Erlbaum Associates, Publishers, London.

Langley, P., and Sage, S. (1984). Conceptual Clustering as Discrimination Learning. *Proceedings of the Fifth Biennal Conference of the Canadian Society for Computational Studies of Intelligence.*

Lebowitz, M. (1983) Generalization from Natural Language Text. In *Cognitive Science*, **7**, 1, 1-40.

Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM. In *Machine Learning*, **2**, 2, 103-138.

Michalski, R. S., and Stepp, R. E. (1983). Learning From Observation: Conceptual Clustering. In *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Michalski, R. S. (1986). Understanding the Nature of Learning: Issues and Research Directions. In *Machine Learning II: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Minsky, M. (1975). A Framework for Representing Knowledge. In *The Psychology of Computer Vision*, P. Winston (Ed.), McGraw-Hill, New York.

Quinlan, J. R. (1983). Learning Efficient Classification Procedures and their Application to Chess End Games. In *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Quinlan, J. R. (1986). The Effect of Noise on Concept Learning. In *Machine Learning II: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Rosch, E. (1973). On the Internal Structure of Perceptual and Semantic Categories. In *Cognitive Development and the Acquisition of Language*, T. E. Moore (Eds.), Academic Press, New York.

Rosch, E. and Mervis, C. B. (1975). Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology* 7 573-605.

Rosch, E., Mervis, C. B., Gray, W., Johnson, D., and Boyes-Braem, P. (1976). Basic Objects in Natural Categories. *Cognitive Psychology* 7 573-605.

Rosch, E. (1978). Principles of Categorization. In *Cognition and Categorization*, E. Rosch and B. B. Lloyd (Eds.), Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Schank, R. C. and Abelson, R. P. (1977). Scripts, Plans, Goals and Understanding. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Stepp, R. E. III. (1984). Conjunctive Conceptual Clustering: A Methodology and Experimentation. *Doctoral Dissertation*, University of Illinois at Urbana-Champaign.

Stepp, R. E. III, and Michalski, R. S. (1986). Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects. In *Machine Learning II: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Vere, S. A. (1975). Induction of Concepts in the Predicate Calculus. *Proceedings of the Fourth International Conference on Artificial Intelligence*, IJCAI, Tbilisi, USSR.

Winston, P. H. (1986). Learning by Augmenting Rules and Accumulating Censors. In *Machine Learning II: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann Publishes, Inc., Los Altos, CA.

Requirements for Computer Aided Knowledge Acquisition in an Ill-Structured Domain: A Case Study

Jürgen Herrmann
Helmut Franzke

Universität Dortmund
FB Informatik I
Postfach 50 05 00
4600 Dortmund 50
F. R. Germany

ABSTRACT

This paper deals with the acquisition of knowledge for a real-word problem: chip-architecture planning, one of the fist steps of IC design. The difference between chip-architecture planning and "typical problems" being addressed by machine learning programs is pointed out. An interactive acquisition component called LEDA that can be integrated into a design assistant (a knowledge-based system that offers the IC designer a set of features and functions that enable him/her to colloaborate with it efficiently [1]) is proposed.

1. Introduction

In recent years the interest in knowledge acquisition for expert systems has grown significantly. The usefullness of machine learning techniques for this tedious and time-consuming task has been demonstrated by the development of several promising systems. The goal to ease this task by minimizing the expert's part in knowledge acquisition is challenging. For different reasons only few research has been concerned with the acquisition of knowledge for complex real-world tasks in an ill-structured domain. To address this gap this paper deals with a complex and relatively poorly understood task that is part of the IC design process: chip-architecture planning.

2. Chip-Architecture Planning

The Task

The design of integrated circuits is an extremely complex task. To receive the final chip layout several abstraction levels and synthesis steps have to be processed. Functional, structural and geometrical aspects of the design artifact are considered, multiple optimization goals are presued.

During chip-architecture planning (typically one of the first steps of IC design) a structural description is synthesized from an abstract functional one. For the functional description a high-level hardware desciption language like DACAPO [2] is used to describe the behaviour of the target IC by means of a concurrent algorithm. In the resulting structural descripition the design artifact is represented as a netlist of

"register-transfer" building blocks (e.g. registers, ALUs, control units, decoders, busses etc.). During the planning procedure the elements of the algorithm (variables, constants, operators, control structures etc.) are mapped onto these building blocks. This task is guided and constrained by several optimization goals: Mimimal overall chip area, maximal speed, good testability, miminal design costs etc.
For the case study described here a part of this complex task has been analyzed with some detail: The allocation of the basic elements of a datapath, namely ALUs, registers, muliplexers and busses. During this task variables and constants are mapped onto registers, operators are mapped onto (usually more than one) ALU. For the assignment of values busses are used. (A simple bus can degenerate to a dedicated connection.) The subset of DACAPO that is considered in this paper is called "Mini-DACAPO".

Usually more than one elements of the algorithm can be mapped onto one data path element. E.g. for various data transports busses can be used, multiple operations can be performed by busses. On the other hand it can be feasible to create more than one register transfer element for an element of the algorithm (hardware doubling) to increase the speed of the aimed chip. To guide this allocation an analysis of the expressions and assignments of the algorithm (e.g. life time analysis for variables) is done. In the following the problem solving procedure for chip-architecture planning shall be explained by use of an example.

Example

For the following simple hardware description in DACAPO an allocation shall be performed:

```
line    DACAPO code
-------------------------
(1)     seqbegin
(2)       c:=a+b;
(3)       a:=d;
(4)       d:=c+b;
(5)     end
```

The seqbegin/end control structure expresses that the enclosed statements are evaluated sequentially. (In DACAPO also concurrency and parallelism can be described.) For the statement in line (2) three registers (for the variables a,b, and c), one ALU (for the addition) and three busses are allocted.

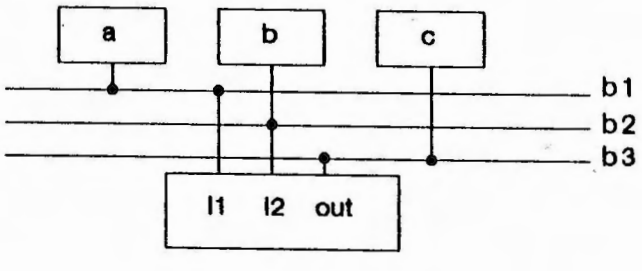


Figure 1: Hardware for the assignment in line (2)

To realize the assignment in line (3) an additional register and an additional bus is allocated. Now register a has to be connected to two busses. Because of that a multiplexer is inserted.

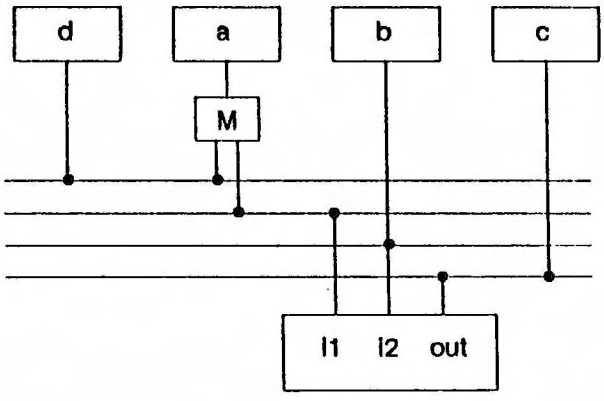


Figure 2: Hardware for the assignments in line (2) and (3)

For the third assignment a new ALU and a multiplexer for register c are allocated.

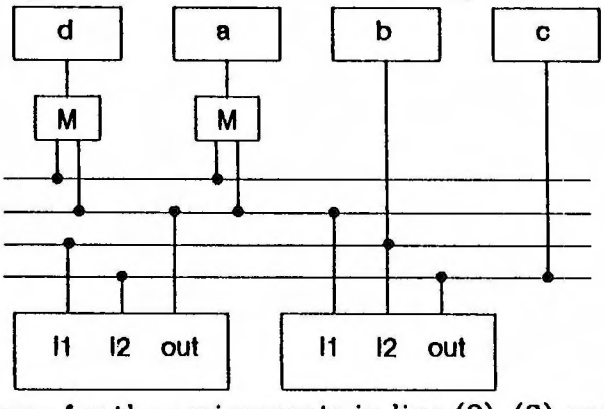


Figure 3: Hardware for the assignments in line (2), (3) and (4)

The result shown in figure 3 is not optimal. There are more ALUs and busses than needed. The additions in line (2) and (4) are performed sequentially. So the same ALU can perform both of them. An optimized allocation is shown in figure 4.

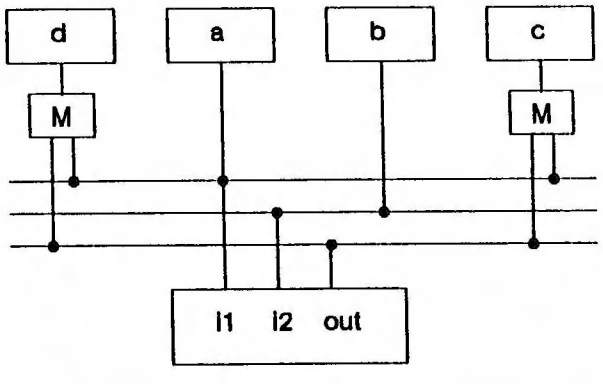


Figure 4: Optimized hardware for the assignments in line (2), (3) and (4)

Why is Knowledge about Chip-Architecture Planning Difficult to Acquire?

As mentioned before chip-architecture planning is different from typical "machine learning problems" (i.e. those tasks that are addressed by most learning systems). In the following we point out these differences.

- *Chip-architecture planning cannot be described adequately by state space search*
  Most systems that learn production rules deal with problems that can be solved adequately by state space search [3,4,5]. Randell writes in [3]: "Just about any problem

can be cast as a search problem." The representation of chip-architecture planning as a search problem with a single state space is problematic. State-spache search requires a stepwise transformation of the problem of interes into a solution. The intermediate states form new (easier to solve) problems. This means that the set of goal states must be a part of the problem space. This condition does not hold for chip-architecture planning. There are two different spaces to consider (functional vs. structural descriptions) that have different primitive elements. There are no intermediate states that form a new problem and are clase to a goal state.
Most of the operators for chip-architecture planning operate in both spaces simultaniously. For example during the allocation of a register for a variable that belongs to the algorithmic hardware description it has to be tested if an already existing register can be used for this purpose. So as well the domain as the range space are analyzed during the execution of this operator. There are also operators that work only in one of the two spaces. An operator that works in the domain space and manipulates the Mini-DACAPO source code is the shifting of an invariant expression from inside a while statement before it. Hardware doubling e.g. for an ALU is an operator that manipulates solely the range space of chip-architecture planning.

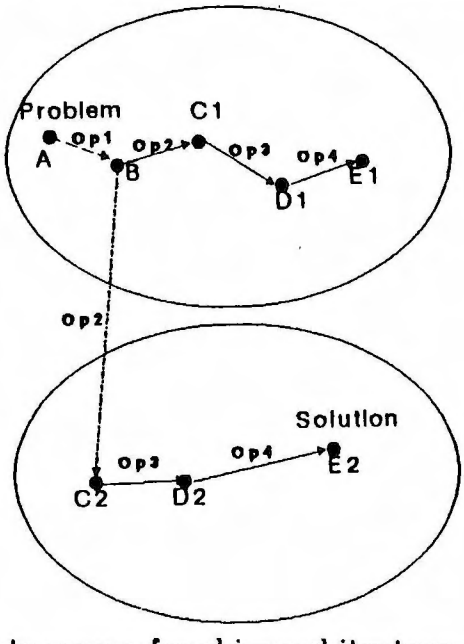


Figure 5: The two state spaces for chip-architecture planning

- *The operators for chip-architecture planning are not simple*
  To allocate hardware for a single assignment there are several "micro operators" to perform. Registers, ALUs and busses must be allocated, connections have to be created. These micro operators depend on each other and have to be executed in certain sequences. The elementary steps cannot be handles separately because only certain combination of them form meaningful design plans (macro operators). Only after the execution of a whole design plan consistant state of the range space is reached, i.e. only then the elements of the register transfer description are connected correctly to each other.
  There are multiple meaningful combinations of micro operators that can form a macro operator but a relatively limited number of micro operators. Because of that the set of micro operators can be put into an acquisition system in advance. In contrary the design plans must be acquired by use of the system.
- *Chip-architecture planning is an optimization problem*
  As mentioned before there are several (partly contrary) optimization goals to consider. There is e.g. a tradeoff between speed and area of the aimed chip. The way this tradeoff is to handle depends among other things on the specified constraints for the artefact and the properties of the used IC technology. Because of these various dependencies a formalized declarative description of the goal states in the range

space is not available. There is no complete or well-structured domain theory for chip-architecture planning. Because of that it is not possible to test automatically, if acquired knowledge leads towards a goal state.

- *The is no weak problem solver for chip-architecture planning*
  Only a few of the macro operators for chip-architecture planning are currently known. A formal description of the goal states does not exist. For that reason a solution for a chip-architecture planning problem cannot be found by extensive search automatically.
  Many existing learning systems (e.g. [3,4,5,6]) base on a weak problem solver and learn knowledge incremetally that improves the problem solving performance.
- *The separation of implementation and strategy rules is unsuitable for chip-architecture planning*
  In the LEAP system [7] that deals with the later stages of the IC design process this separation is used. LEAP acquires rules that represent legal implementation operators and do not contain any strategic knowledge. In this way no optimization goals have to be considered during the learning procedure. For chip-architecture planning this separation cannot be made. Here the objects of interest are the (macro) planning operators. Their composition from micro operators already depends on strategic aspects. So the learning of macro operators is influenced by strategies for chip-architecture planning.
- *Chip-architecture planning is influenced by aspects that are represented implicitly in the problem description*
  E.g. the life time of a register in not described explicitly in a Mini-DACAPO program. It has to be extracted by an analysis program. Because of the large number aspects or interest it is not feasible to add information about all aspects to each state description of the problem solvinng procedure.

Because of the above listed reasons the existing methods for learning of rules from examples [4,6,7] cannot be applied to chip-architecture planning. Instead of that we propose an *interactive acquisition program* that is tailored for this specific task and is able to perform single generalization steps independently.

3. Conception of an Acquisition Tool for Chip-Architecture Planning

This chapter describes the working principle of the proposed acquisition component for chip-architecture planning.

As mentioned before an acquisition program for chip-architecture planning has to work interactively. On the other hand experts for IC design are seldom and expensive. To cope with these difficulties LEDA is organized as a learning apprentice system [7]. This means that it offers features to the expert that he/she uses to plan his/her chip-architecture "manually". In this way examples are made available for LEDA. LEDA generalizes this information. Because automatic criticism of the results is not possible the program asks the expert to criticize the generated knowledge.

Figure 6 shows the structure of the LEDA system introduced in this paper.

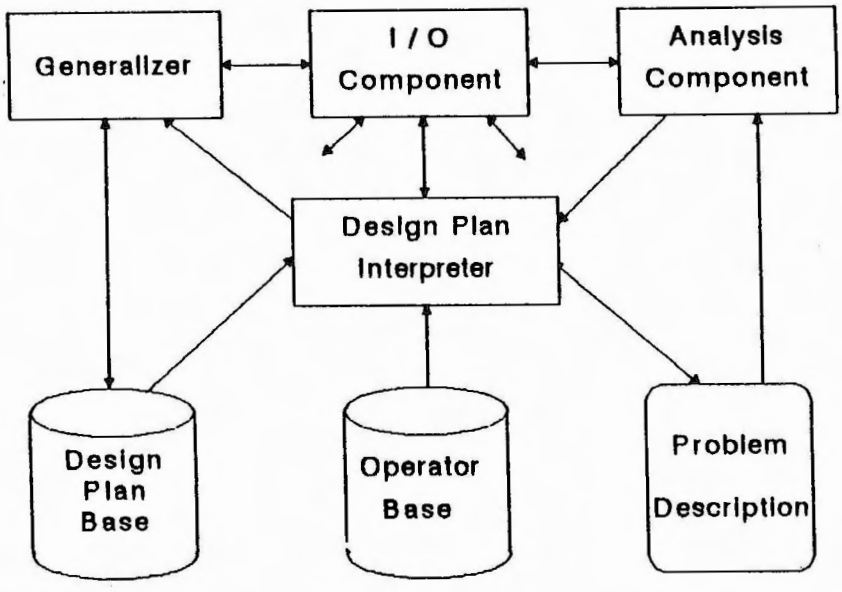


Figure 6: Structure of the LEDA system

The ***problem description*** describes the currrent state of the considered example problem. It consists of a Mini-DACAPO program and a description of the allocated hardware structure. The *I/O component* forms the interface to the user. Using this module he/she can observe all parts of the system, answer questions LEDA generates, deliver examples and change the knowledge represented in the different parts of the system. The ***operator base*** contains all micro operators the system knows. In the ***design plan base*** the acquired macro operators are stored. The ***design plan interpreter*** stepwise applies the micro operators to the problem description. The evaluation of the design plans is guided by the results of the ***analysis component***. The ***generalizer*** puts the design plans that are acquired in cooperation with the user into the design step base. It applies generalization operators to the design plans.

LEDA's cooperation with the user is organized in the following way: At the beginning of a session the example to be solved exists in the problem description in form of a Mini-DACAPO program. The user marks the first statement hardware shall be allocated for. The design plan interpreter looks for a suitable macro operator in the design plan base. If it finds one that is accepted by the user it is applied to the problem description. Otherwise the user him-/herself delivers a new sequence of micro operators that allocates hardware for the considered statement. Each micro operation is evaluated immediately and the resulting modified problem description is shown to the user. The analysis component is applied to each (intermediate) state if necessary. It delivers additional information about the problem decription. The completed sequence of micro operators is inserted into the design plan base and the generalizer is activated. It looks for another plan that is similar to the new one in the design plan base. If this search is successful it tries to construct a new more general design plan that can substitute the these two plans. The generalization steps performed by LEDA are critizied by the user.

As mentioned before LEDA's knowledge is represented as episodes of micro operators. Knowledge about the various aspects of chip-architecture planning can be described with these macro operators. LEDA offers the following possibilites for the acquisition and refinement of episodes.

1) Hardware allocation for a single statement
2) Generalization of design plans
3) Inclusion of alternatives into a design plan
4) Integration of hardware for a single statement into a given context

3.1. Hardware allocation for a single statement

The most simple design plans LEDA can acquire deal with single assignments. Each plan allocates hardware for a specific assignment without considering the context of the statement. Plans of this kind can be applied at the beginning of a chip-architecture planning procedure. The following example demonstrates the interactive construction of a design plan for a simple assignment.

Example
For the statement "a := b + 1" hardware shall be allocated. To start the allocation the user puts the statement into the system:

```
=> (a := b + 1)
```

The sign "=>" denotes that the current line is delivered by the user. Let us assume that LEDA has no design plan for this assigment at the moment. Because of that the user presents a sequence of micro operators that performs the allocation. The first three micro operators allocate the registers needed:

```
=> (alloc-reg a) ;Map the variable a on a register
=> (alloc-reg b) ;Map the variable b on a register
=> (alloc-reg 1) ;Map the variable c on a register
```

LEDA's design plan interpreter performs these operators. The registers receive the names of the corresponding variables and constants. The I/O displays shows the hardware that has been allocated by now to the user.

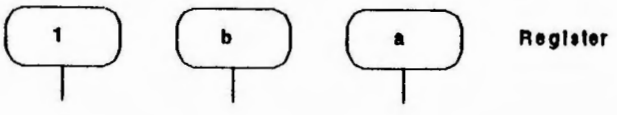


Figure 7: Intermediate state of the allocation (1)

For the "+" operator the user orders the allocation of an ALU:

```
=> (alloc-alu + alu1) ;Realize the operator "+" with am ALU
                        named alu1
```

This leads to the following structure

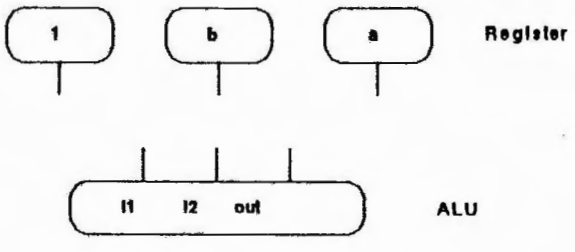


Figure 8: Intermediate state of the allocation (2)

For the communication of the hardware elements allocated above busses are needed:

=> (alloc-bus b1) ;Allocate a bus with the name b1
=> (alloc-bus b2) ;Allocate a bus with the name b2
=> (alloc-bus b3) ;Allocate a bus with the name b3

Now all necessary components have been allocated.

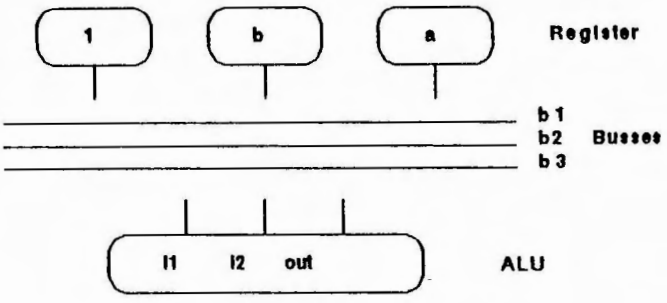


Figure 9: Intermediate state of the allocation (3)

To complete the design plan the user describes the connections of the hardware elements. He gives the following micro operators:

=> (connect b1 a) ;Connect register a with bus b1
=> (connect b1 alu1-out) ;Connect b1 with the output of alu1
=> (connect b2 b) ;Connect register b with bus b2
=> (connect b2 alu1-in2) ;Connect b2 with the second input of alu1
=> (connect b3 1) ;Connect register 1 with bus b3
=> (connect b3 alu1-in1) ;Connect b3 with the first input of alu1
=> (ready)

"(ready)" denotes the end of the design plan. The I/O component displays the resulting structure to the user.

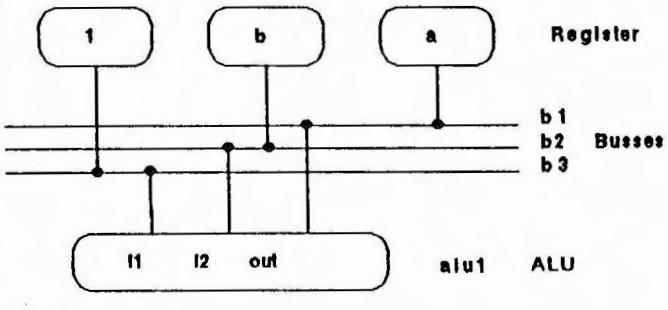


Figure 10: Result of the allocation

Internally LEDA represents this structure in form of a list:

{(reg a) (reg b) (reg 1) (alu alu1) (bus b1) (bus b2) (bus b3) (conn b1 a) (conn b2 b) (conn b3 1) (conn b1 alu1-out) (conn b2 alu1-in2) (conn b3 alu1-in1)}

3.2. Generalization of design plans

The acquired design plan is very specific. It only can be applied to exactly one statement. So it is of limited use currently. It must be generalized.

For other assigments like e.g. "c := b + 1" that are very similar to the one considered above the design plan does not work. A correct plan for this assignment would be nearly identical to it. To avoid the coexistence of nearly identical design plans in LEDA's design plan base the generalizer is activated each time a new design plan is inserted into the design plan base. The generalizer tests if a similar design plan exists already. To detect similarities of design plans LEDA uses concept hierarchy trees. The trees can be derived from the grammar of Mini-DACAPO.

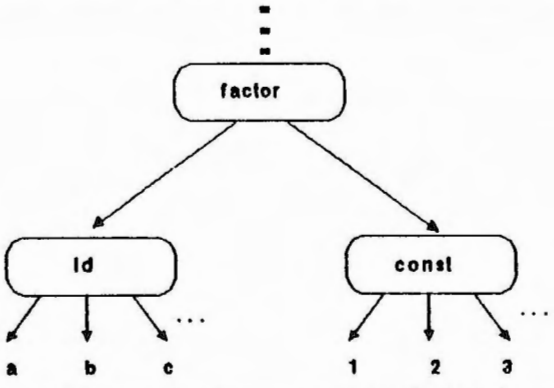


Figure 11: A concept hierarchy trees used by LEDA (excerpt)

LEDA assumes that two assignments are similar if they consist of the same number of terms and if the in pairs corresponding terms are identical or have the same father in the concept hierarchy tree.

Examples
(a := b + 1) and (c := b + 1) are similar.
(a := b) and (c := d) are similar.
(a := b + 1) and (a := b - 2) are similar.
(a := b + c) and (a := b + c + 5) are not similar.

If LEDA finds a pair of similar assignments it tests if the similarity holds also for the two corresponding plans. In this case the two plans can be generalized by use of the climbing concept hierarchy generalization rule, a typical generalization operator for concept acquisition from examples [8]. The generalized statement and design plan are constructed by substituting each pair of similar but not identical terms by the corresponding father node in the concept hierarchy tree.

Example
(a := b + 1) and (c := b + 1) are generalized to (<id> := b + 1).

The corresponding generalized design plan can be applied (matches on) the set of all statements of the form "<id> := b + 1". It is still more specific than necessary. E.g. the operator "+" can be generalized to <arithmetic-op>. As a result of several generalization steps the design plan can be generalized to match on all statements of the form

"<id> := <factor> <arithmetic-op> <factor>"

As mentioned before it is not possible to test automatically if the generalized design plan reaches a correct goal state. Because of that each single generalization step is criticized by the user. If necessary he/she resets faulty generalization steps (e.g. "over generalizations").

3.3. Inclusion of alternatives into a design plan

The generalized design plans described above are still relatively specific and unflexible. Depending on the context of an allocation different alternatives can arise during the execution of a design plan. Unfortunately in many cases the selection of the right one cannot be made before the execution of the plan because the necessary information is lacking at that point. To cope with this problem design plans can be split up. For this purpose the ***select operator*** is used. Utilizing this operator the user can refine an existing design plan and make it applicable to slightly different situations in this way. The use of select operators in design plans has another advantage. Design plans that differ only in one micro operator can be merged in this way. This is another way to get more general design plans.

Example
The user wants to allocate hardware for the statement "a := b + 1". LEDA retrieves a design plan for this statement (see above) that could be applied but is not well-suited for the current state of the chip-architecture planning procedure. To modify this plan the user orders a stepwise execution of the micro operators. After the execution of the first three steps (alloc-reg a), (alloc-reg b), (alloc-reg 1) the user intervenes. He wants to use an existing ALU for the addition. So he selects the micro operation (overlay "+" alux). LEDA detects the difference to the original plan and asks for the condition the selection of the alternative depends on. The user answers:

```
=> (exist-free-alu-for "+" alux) ;There is an ALU named alux that
                                 ;is not used during the execution
                                 ;of the current statement
```

Now LEDA can substitute the original micro operator by the following one:

```
(SELECT (exist-free-alu-for "+" alux)
   TRUE  (overlay "+" alux)
  FALSE (alloc-alu alu1)
 )
```

For the evaluation of the condition an analysis of the current state of the planning procedure is necessary. This is delivered by the analysis component that evaluated besides other aspects the life time of registers, ALUs and busses. Methods for this analysis are described in [9].

The select-operator enables the user to modify and refine existing design plans in an easy way. Instead of delivering a complete alternative design plan for a slightly different situation he/she can include the alternative problem solving step(s) into the existing plan. As mentioned before the quality of the chip-architecture plan depends on many aspects. The selected operator makes the inclusion of specific conditions (that can be evaluated by the analysis component) into a design plan easy. It would not be feasible to analyze all aspects of a chip-architecture plan for each intermediate state of the planning procedure. This would lead to a combinatorical explosion. The aspects a select operator depends on are evaluated only for the specific situation this information is needed for.

3.4. Integration of hardware for a single statement into a given context

The design plans described so far have no regard to the context of the hardware elements they allocate. To enable the allocation of hardware for a complete Mini-DACAPO program the plans have to be refined accordingly.

Example
For the following small program hardware shall be allocated.

```
line    DACAPO code
-------------------------
(1)     seqbegin
(2)       c:=a+b;
(3)       a:=b+1;
(4)     end
```

The user causes and controls the execution of an existing design plan for the statement in line (2). For this statement currently there exists no hardware that must be considered. So the plan can be executed without changes. The resulting hardware is shown in figure 12.

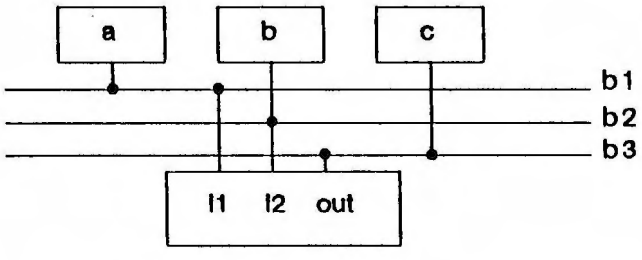


Figure 12: Hardware for the assignment in line (2)

For the statement "a := b + 1" the design plan described above exists. To consider the now existing context (hardware for line (2)) the plan must be refined. The user causes a stepwise execution and inserts modifications using the select operator. E.g. the micro operation (alloc-reg a) is not needed in this context because a register for the variable a exists already. Instead a (noop) operation is needed. This leads to the following select operation that substitutes the original micro operator.

```
(SELECT (exist-reg a)
   TRUE  (noop)
  FALSE (alloc-reg a)
 )
```

The other micro operators that are influenced by the context are changed accordingly. The final hardware for the small program is shown in figure 13.

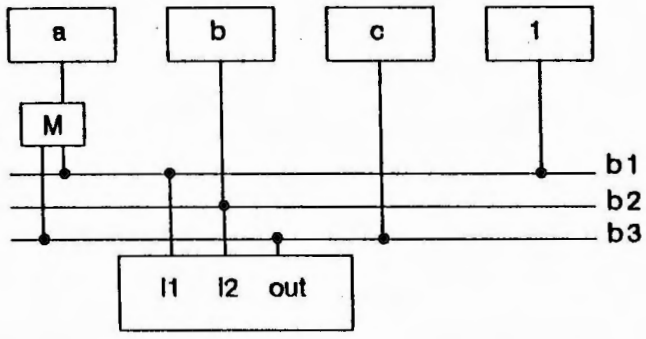


Figure 13: Hardware for the small example program

4. Further Issues

- The design plans described above only deal with assignments that occur in a seqbegin/end block. Besides that there are other contexts for an assigment (conbegin/end block, while statement, if-then-else statement ...) that require different hardware allocations. So the context for an assignment in the Mini-DACAPO program must be considered in the design plans.
- There are diverse aspects the quality of a chip-architecture plan can depend on. Many of them have to be extracted from LEDA's problem description by the analysis component. Not all of them will be known at the beginning of experiences with a LEDA prototype. So the system must enable the user to specify new analysis aspects during the use of the system. This means that it must support a dynamic extension of the analysis component.
- In the current concept for LEDA only the climbing concept hierarchy tree generalization operator is considered. The inclusion of other generalization techniques is desirable to make the acquisition preocedure more effective.

5. Conclusions

In the case study described here specific properties of the complex design problem chip-architecture planning have been described. The study is related to the development of an expert system for chip-architecture planning [10] that is currently developed at the University of Dortmund, Lehrstuhl Informatik I (Prof. Dr. B. Reusch).

We have pointed out that the automatic acquisition of knowledge for this task is not possible. As an alternative we have proposed the *interactive acquisition program* LEDA. LEDA's concept bases on the following principles:

- The design plan is the central structure of LEDA's knowledge base. The acquisition of single isolated micro operators is not feasible for chip-architecture planning.
- The generalization of the acquired design plans is initiated by LEDA. The user only needs to criticize (and reset if necessary) the proposals LEDA delivers.
- The knowledge acquisition procedure is charcacterized by the interactive solution of example problems. This reflects the widely accepted thesis that experts rather can demonstrate their knowledge than formulate it abstractly.
- The system has an significant amount of domain knowledge (knowledge about analysis of chip-architecture plans and mirco operators for chip-architecture planning). Because of that the user needs not formulate the "basis building blocks" of his/her knowledge. The domain knowledge enables the system to evaluate the acquired knowledge and show its effect on the considered example to the user immediately.

- The initial design plans are relatively simple an partly incomplete. By considering different contexts for a single assignment the expert can refine the design plans in interaction with LEDA easily.

References

[1] J. Herrmann, B. Reusch
Combining Expert Systems and Machine Learning in CAD-Systems for Micro-Electronics
Proc. of Int. Workshop on AI-Applications to CAD-Systems for Electronics, 1987

[2] DACAPO-Manual
Edition '87, Dosis GmbH, Dortmund

[3] L. Randell
Conceptual Knowledge Acquisition in Search
in: Computational Models of Learning, L. Bolc (ed.)
Springer Verlag 1987

[4] T.M. Mitchell, P.E. Utgoff, R. Banerji
Learning By Experimentation: Acquiring and Refining Problem-Solving Heuristics
in: Machine Learning, R.S. Michalski, T.M. Mitchell and J.G. Carbonell (eds.)
Tioga Publishing 1983

[5] A.L. Samuel
Some studies in machine learning using the game of checkers
in: Computers and Thought, E.A. Feigenbaum, J. Feldman (eds.)
McGraw-Hill 1963

[6] B.W. Porter
Learning Problem Solving
PhD-Thesis, University of Irvine 1984

[7] R.S. Mitchell, S. Mahadevan, L. Steinberg
LEAP: A learning apprentice for VLSI design
Proceeding of IJCAI 1985

[8] R.S. Michalski
A Theory and Methodology of Inductive Learning
in: Machine Learning, R.S. Michalski, T.M. Mitchell and J.G. Carbonell (eds.)
Tioga Publishing 1983

[9] R. Brück, R. Klomps, J. Schütz
Analysis of Behavioral DACAPO Descriptions
CHDL 1987

[10] K.-H. Temme and R. Brück
Chip-Architecture Planning Based on Expert Knowledge
Proceedings of International Workshop on Artificial Intelligence for Industrial Applications
Hitachi City, Japan, May 1988

WASTL: An Approach to Knowledge Acquisition in the Natural Language Domain

Roman M. Jansen-Winkeln
Universität des Saarlandes
Federal Republic of Germany
roman@fb10vax.informatik.uni-saarland.dbp.de

Abstract

Characteristic of natural language systems is their use of numerous knowledge bases. Acquisition of knowledge has to accommodate the special demands of natural language as well as the complexity of the developed systems. The following paper presents concepts dealing with integrated knowledge acquisition for XTRA – a natural language access systems for expert systems. Integration occurs through the application of a common specification methodology for application and knowledge acquisition system – the KADS methodology by Breuker et al. Given this foundation, it is possible to define application and knowledge acquisition systems that allow for the mutual use of modules and an extension of additional knowledge acquisition methods.

A special advantage in using a common specification is that the selection of knowledge acquisition methods can, for the most part, be automated. Knowledge acquisition systems can be applied successfully by evaluating the (fail) reactions of the application system, creating the basis for unifying powerful, but often only partially suitable approaches – inductive learning, concept classification and concretion, paraphrasing etc. – under a common and in part natural language surface.

1 Introduction

WASTL (WisssensAkquisitonSTooL) represents the planned knowledge acquisition tool for XTRA – a natural language access system to expert systems. XTRA is currently being developed at the Universität des Saarlandes, West Germany. It communicates with the user in a natural language (German), extracts data relevant to the expert system from his/her natural-language input, answers user queries as to terminology and provides user-accommodated natural-language verbalizations of results and explanations provided by the expert system. XTRA is being developed independently of any specific expert system One research goal is the adaptation of this access system to different expert systems. For this reason, research on knowledge acquisition, next to the examination of linguistic and AI-methodical question formulation, plays an important part. The goal of knowledge acquisition for the immediate future will be the adaptation of domain-dependent knowledge components to newly interfaced expert systems. Thereafter, it will be employed to support the construction and integration of new knowledge bases and inference components, as well as to maintenance domain-independent components of the system.

1.1 Knowledge Bases in Natural Language Systems

Modern natural language systems use a number of different knowledge bases. For example: [1]

- General background knowledge such as word and idiom lexicon, morphological, grammatical, and conceptual knowledge, or, rules pertaining to dialog and text construction.
- Domain-dependent knowledge such as referential knowledge, a-priori user models or special dialog strategies.
- Dialog- and text-related knowledge such as syntactical and semantical memory, user models or focus.

Knowledge bases are generally classified according to their relationship to processing steps within the natural language system (e.g. syntax, semantics, pragmatism) as well as to their state of general validity. Relevant for knowledge acquisition are mainly domain-dependent knowledge bases and basic background knowledge. On the other hand, the natural language system itself is normally responsible for the construction and management of dialog- and text-related knowledge bases.

1.2 Knowledge Acquisition Approaches

Knowledge acquisition approaches in the field of natural language systems can be divided into three large groups.

1.2.1 Formalism-Specific Support

The first group contains systems that support the development of knowledge bases in a formalism-specific manner. Within these systems, the linguistic workbenches are the most

[1] See W.Wahlster [Wah82].

popular. Examples are Kaplan's LFG-Workbench [KB82], Karttunen's D-PATR environment [Kar86], or the TAG-Development-Environment designed in Saarbrücken [Sch88] These represent complex software systems that give a linguist the capability of transforming his technical grammar knowledge into executable grammar on his own.

This group also contains systems that support the construction of knowledge bases within representation formalisms which are preferably employed within natural language systems. For example, the KREME-system developed at BBN [AB86], a Knowledge Editing Environment basically suited to frame representation languages similar to Brachman's KL-ONE [Bra82]. In natural language systems, these representation languages are mainly used to model semantical knowledge. Common to all of these systems is that the man-machine interface is tailored to the technical knowledge of a specific type of expert. Typically, these systems do not contain information concerning application and application demands of the respective knowledge base being constructed.

1.2.2 System-Specific Knowledge Acquisition

The second group is comprised of knowledge acquisition programs developed for specific natural language systems. Prototypical is the semantic acquisition in TELI presented by Ballard and Stumberger [Bal86,BS86] and the concept acquisition in TEAM developed by Grosz et al [GAMP85]. Characteristic of these systems are powerful, special purpose inferences tailored to the particular natural language access system for which knowledge is acquired. A disadvantage though is the stereotypical questioning of the user and the incapability of utilizing additional user information. Also, the user is generally not able to influence the modeling aspects of the knowledge acquired.

1.2.3 Machine Learning Techniques

Finally, techniques out of machine learning research can be used for the automated construction of knowledge bases in natural language systems. To a great extent, these approaches deal with syntactical aspects of natural language, stressing those phenomena upon which formally sound theories such as grammars and automata can be applied. For example, several projects relating to morphological knowledge contribution (e.g. [Wot84]), Berwick and Pilato's contribution "Reversible Automata and Induction of the English Auxiliary System" [PB85] or my own work regarding the induction of grammar rules from examples [Jan87]. In the field of semantics, late research done by James Pustejovsky dealing with the acquisition of conceptual structures is especially interesting [PB87]. He is designing a computational model of the acquisition of thematic relations and case roles by developing a representation theory and using learnability theory as a metric for evaluating the model.

1.2.4 Discussion

Common to all of the approaches mentioned above are their limitations, regardless of whether the supported formalism, the supported systems and their applications, or the supported aspects of natural language is meant. In addition, systems within the first and third group cannot, as a rule, relate to those components of the supported natural language system which are not directly effected by the acquisition.

Integration is, in general, a weak spot within current natural language systems. One exception is the IRACQ system – the knowledge acquisition component of the natural

language interfaces IRUS and JANUS [WAH*87]. From sample clauses supplied by the user, IRACQ suggests to the user the predicates to be used in semantic interpretation, assuring consistency with the model of the domain which is represented in NIKL [Mos83] and uses the full power of the IRUS parser and interpreter to interpret the sample sentences.

Moreover, the approaches mentioned above seldomly stress the actual importance of the numerous knowledge bases within natural language systems. One of the few that examines the mutual relationship between knowledge bases systematically is IWAS, the knowledge acquisition component of the HAM-ANS system [BP85]. Within this approach, knowledge element related, knowledge base local, and knowledge base overlapping constraints are defined with respect to completeness and consistency of knowledge bases in general. Changes within a knowledge base are described through generic operations on knowledge elements and carried out through formalism-specific instances of these operations.

The two latter systems present ideas which influenced the design of the knowledge acquisition tool WASTL. The idea of eliciting new knowledge units from sample clauses was taken over from IRACQ. The approach was generalized and under the name "*example-driven*" knowledge acquisition it forms the framework for knowledge acquisition through WASTL. The hierarchical overhead of knowledge base descriptions and constraints in the IWAS approach is replaced by a specification of both, the application and the knowledge acquisition system through a common formalism – the KADS methodology. Due to the easy definition of data and functional interfaces, this specification sets the basis for an integration of application and knowledge acquisition with mutual advantages. Thus WASTL can use XTRA modules for the construction and verification of hypotheses. Many of the abilities inherent in XTRA's knowledge bases can be exploited such as handling of incomplete knowledge structures or consideration of constraints within knowledge bases. On the other hand, this integration allows XTRA to select suitable knowledge acquisition modules whenever required.

After a short survey of the XTRA system in the following section, KADS will be presented in section 3. With the help of examples section 4 describes some of the knowledge acquisition modules, the integration of knowledge acquisition and application as well as the overall framework of example-driven knowledge acquisition.

2 The Natural Language Access System XTRA

Figure 1 outlines the architecture of the XTRA system, characterized by its modular construction and its rigid, essentially sequential flow of control.

The operating procedure of the XTRA system can be sub-divided into three phases: analysis, evaluation, and generation. The analysis phase transforms the written natural language input (expanded through deictic data) into an internal domain-dependent representation – the conceptual knowledge base (CKB). Based upon this representation, interaction with the respective expert system takes place, the result of which also being represented in the CKB. These results provide the basis for the subsequent generation of natural language output and pointing gestures.

Knowledge bases that takes part in the analysis and generation respectively are, for example, the morphological lexicon XTRALEX, the unification grammar XTRAGRAMM (realized within the PATR-formalism) or the functional semantic structure (FSS), a gen-

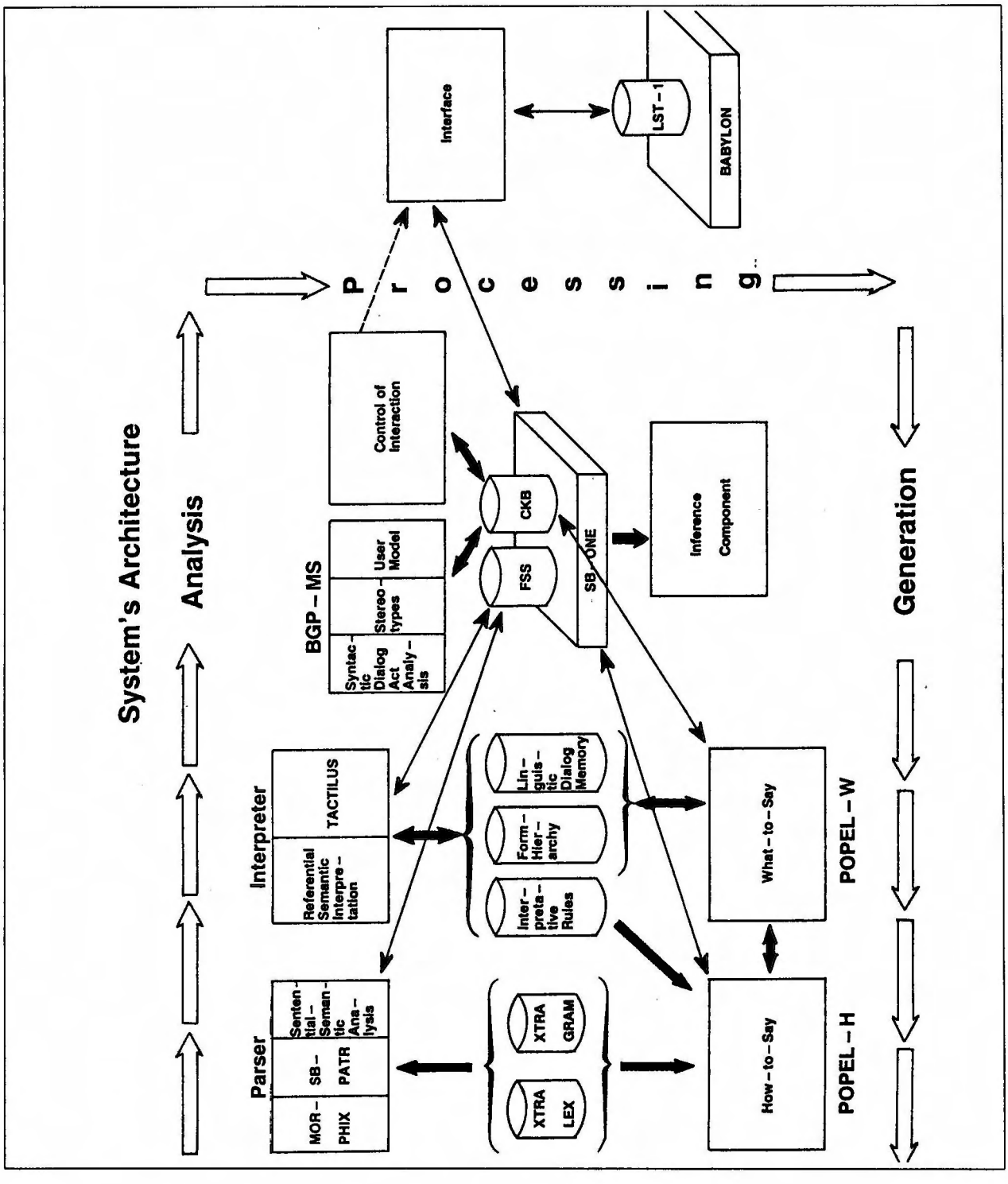


Figure 1: XTRA – Knowledge Bases and their Interaction

eralization of Fillmore's case frames, implemented in SB-ONE, a derivate of KL-ONE[2].

[2]A detailed description of the XTRA project can be found in [AHK*88].

3 KADS-Methodology

For a program of XTRA's complexity, it seems to be necessary to explicitly specify the operating procedure of the system. The KADS methodology, developed within an ESPRIT project by Breuker et al., can be used to solve this specification problem [BWvS*86]. KADS is characterized by the following:

- KADS is tailored to the special language elements and inference methods of knowledge based systems.
- The approach is sufficiently informal in order to guarantee easy handling and quick results.
- The approach is sufficiently formal in order to at least, in part, be operational.
- KADS is based upon an adequate level of abstraction – the epistemological problem-solving level.

The most important function of this methodology is the analysis and functional abstraction of (human) problem-solving, which can be simplified with the help of the formalism of interpretation models. Considering a 5-steps division of problem-solving analysis [3] – knowledge identification, knowledge conceptualization, epistemological analysis, logical analysis and implementational analysis – the interpretation models are thus to be established at the level of epistemological analysis. Four levels of abstraction are differentiated within each interpretation model: the domain, inference, task, and strategic level. The relationships between the individual steps are shown in figure 2.

level	relation	objects	organization
domain level		concepts, relations, structures	axiomatic structure
⇑	describes		
inference level		meta-classes, knowledge-sources	inference structure
⇑	applies		
task level		goals, tasks	task structure
⇑	controls		
strategic level		plans, meta-rules, repairs, impasses	process structure

Figure 2: Levels of Abstraction in KADS

Within this framework, inference structures are especially important. Their basic elements consist of meta-classes and knowledge sources which represent a classification of existing concepts, relations, and inferences on the domain level. Inference structures unambiguously characterize an interpretation model. They represent the connection between the questions of as to *how* knowledge (domain structure) is *when* (task structure) processed. The strategic level allows for switching over between different task structures, being dependent upon certain modalities on domain-specific events.

In [Jan88], it is shown that and to what extent this specification methodology for the XTRA system can be used. Out of an informal specification (see figure 1), a so-called *real world model* (see figure 3) can be developed step-by-step. If possible, individual subtasks therein are decomposed to generic subtasks (see figure 4).

[3] corresponding to the five levels of knowledge representation [Bra82].

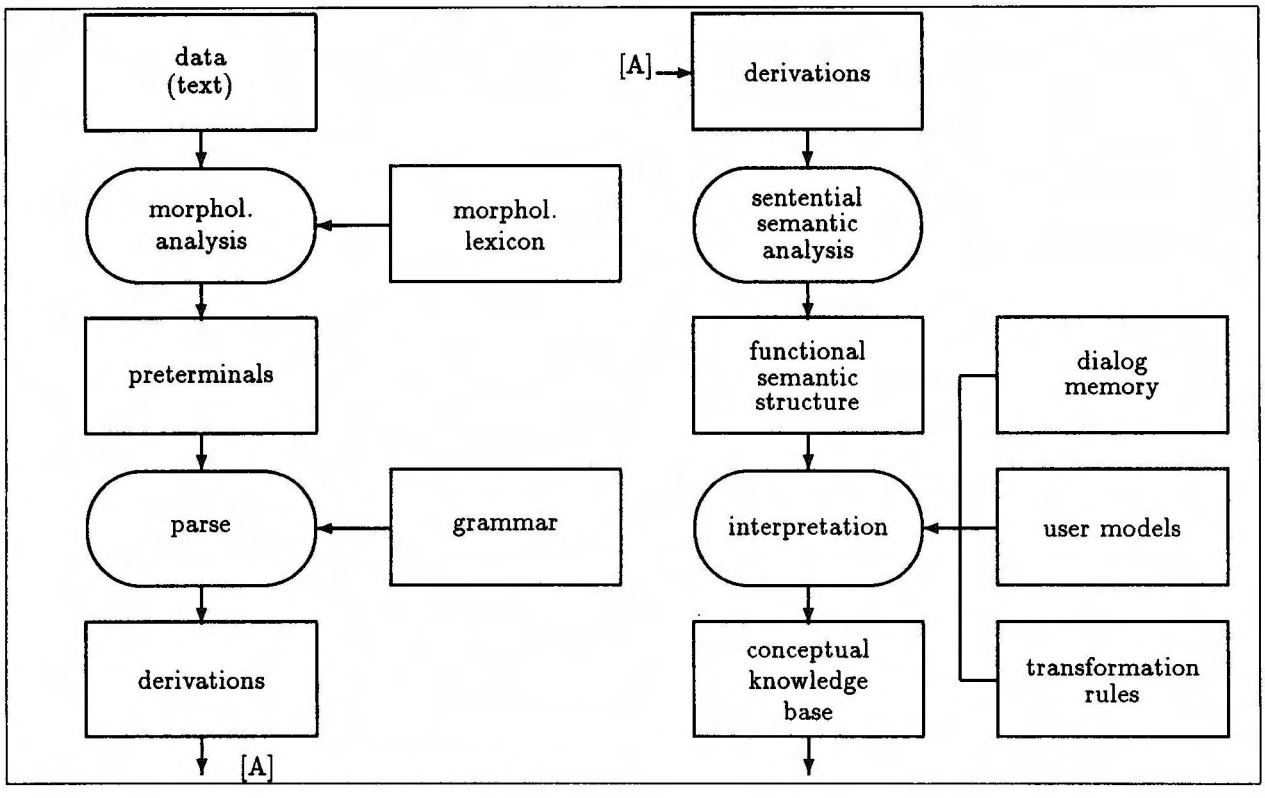


Figure 3: Real World Inference Structure of XTRA's Analysis Phase

This specification procedure only partly corresponds to the original intension of the KADS researchers. Here, a *bottom up* analysis is carried out, but, instead of assuming the expertise of a human being, it tries to model the functionality of an existing program.

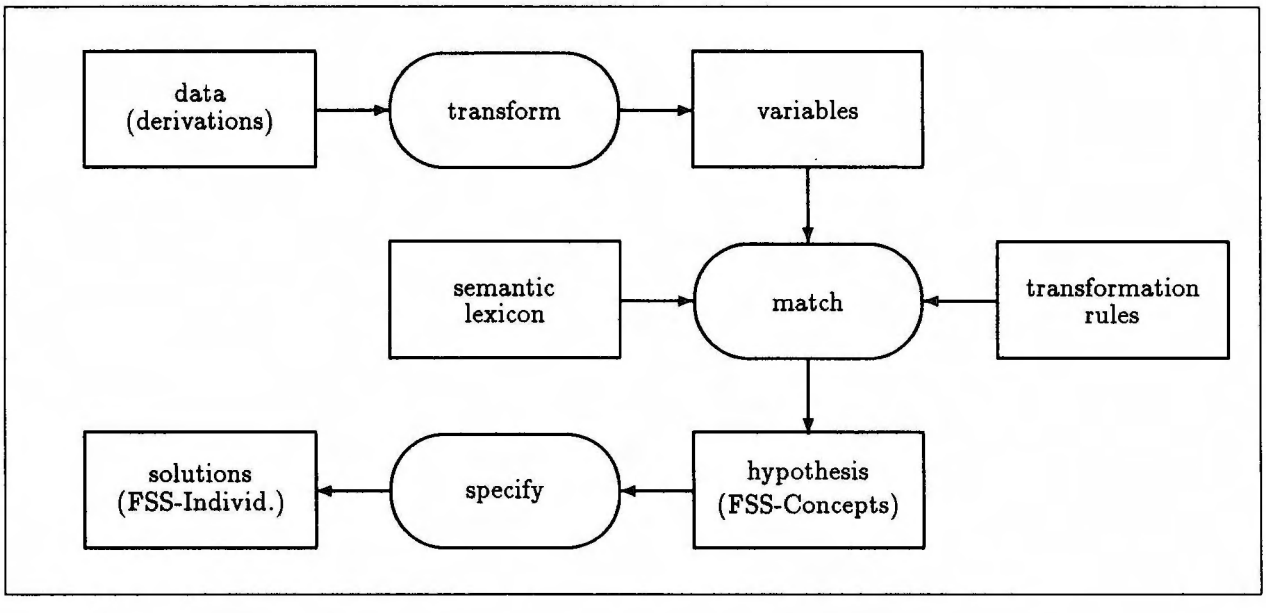


Figure 4: Sentential Semantic Analysis as Heuristic Classification

4 The Knowledge Acquisition Tool WASTL

4.1 Demands on Knowledge Acquisition

Knowledge acquisition in the natural language dialog system XTRA has two main points of emphasis. The first is the *knowledge engineering aspect*, the goal of which is a well-defined interface between linguistic theory on the one hand and AI methodology on the other. The development of structures in order to define, apply, and evaluate a natural language system allow AI techniques to rationally solve linguistic and computer-linguistic problems. Epistemological analysis (see section 3) serves to separate these two levels.

The second point of emphasis deals with the need to expand and update the knowledge bases. For domain-specific knowledge bases, this should be carried out as practically as possible and for the most part, computer-aided. With XTRA – and natural language systems in general - the possibility of using redundancy within a language, the sound theoretical treatment of linguistics, as well as the inherent skill of most users should be utilized in order to gain new knowledge. Primary source of information for WASTL are natural language utterances of the user. Examplary input is used in order to closely examine the operational procedure of XTRA and to check for and correct any incompleteness and inconsistencies within the systems knowledge bases. For this purpose, WASTL makes use of knowledge acquisition modules, specialized on subproblems like acquisition of morphological information, and selected modules of the application system, e.g. XTRA's parser in the following example. These modules will be applied on the user's input and the derived intermediate results. To do this, WASTL requires the possibility to observe XTRA's flow of control as well as the ability to flexibly reorganize it in a problem-specific way. The following examples show that task and strategic level of the KADS formalism provide means to realize these demands.

4.2 A Scenario for Example-Driven Knowledge Acquisition

The imaginary German word "gruspelt" will now illustrate the basic idea of example-driven knowlegde acquisition within the WASTL system. [4]

4.2.1 Syntactical Analysis

Consider the following user input:

> `Peter gruspelt von Dudweiler nach Saarbrücken.`
> (*Peter grusples from Dudweiler to Saarbrücken.*)

Because the system recognizes all of the words except for `gruspelt`, an error will invariably arise in the course of syntactical analysis. Should the knowledge acquisition component be activated in helping to eliminate this error, it will first search for the origin of the error before undertaking corrective measures. If the error is recognized as being an unknown word, the following strategy can be applied:

> The goal is the construction of an entry for the unknown word `gruspelt` within the morphological lexicon of the natural language system. Which knowledge bases in XTRA can help to solve this problem?

[4] The following examples will be given in German with the English translation (as far as possible) in parenthesis.

- the morphological component, containing knowledge in conjunction analysis, vowel mutation reduction etc., and thus being able to supply hypothesis for possible word categories.
- the grammar, which is capable of parsing the input sentence to a grammatically well-formed derivation by locating the appropriate construction out of a set of possible word categories.

These knowledge bases are used separate from their "normal" function within XTRA's flow of control. Given suitable input parameters, they should be able to find an "acceptable" restriction of the search space for possible lexicon entries.

With respect to the example above, this strategy leads to the following results:

1. The application of morphological knowledge gives following hypotheses for the morphological analysis of `gruspelt`:

   Characteristics: endings: "␣" or "`t`", vowel mutation: *false*, participle: *possible*.

   Hypothesis 1 *ending* " ": noun, verb, possessive pronoun, adjective positive used as predicate

   Hypothesis 2 *ending* "`t`": verb; imperative plural, imperfect plural 2, conjunctive 1 & 2 plural 2, present singular 3, present plural 2.

2. Application of the parser to the example sentence: `Peter gruspelt von Dudweiler nach Saarbrücken.` All hypotheses of a word category for `gruspelt` are known to the parser.
   The results are: Number of derivations: 1, Tense: –, Voice: active, Form: finite, Number: singular, Person: 3

3. The renewed application of morphological knowledge allows the following proposal to be made for an entry into the morphological lexicon:

   stem: `gruspel`, *word category*: `verb`, *stem ending category*: `A1`,
   auxiliary verb: `seien`, *participle perfect*: `gegruspelt`

In the last step, the search space had been sufficiently reduced, allowing the hypothesis found therein to be suggested as a possible lexicon entry. Furthermore, the user himself can partake in the processing steps that follow. He then would have to finally decide if the suggested solution is to be accepted and stored in the morphological lexicon. Should he reject this solution and propose another, it should be possible to analyse why his particular solution had not been found. In this way, conclusions can be drawn regarding incompleteness and inconsistency within the domain-independent part of the knowledge bases – in this case grammar and morphological knowledge.

4.2.2 Semantical Analysis

A similar solution is found in semantical derivation, i.e. the analysis steps following syntactical analysis. In XTRA, this means that, with the help of transformation rules and a semantical lexicon, the syntactical derivations (DAGs) are first projected onto indivualizations of the functional semantic structure (FSS). The search within the semantical lexicon

is carried out using the root of the word as a search key. Therefore, in our example, an error would occur while searching for the entry `gruspel` (*to grusple*).
A possible solution to this problem might be the following:

> The goal is the construction of an entry in the semantical lexicon for the word `gruspel`. Which knowledge bases contain the knowledge needed to reach this goal?
>
> - The FSS contains a functional abstraction of all semantically well-formed sentence constructions on a domain-independent level.
> - The semantic lexicon, which contains descriptions of words in a terminology defined by the FSS.
>
> The verb `gruspel` is to be classified as exact as possible using the terminology of the FSS and compared to similar entries within the semantical lexicon.

This strategy allows the verb `gruspel` to be classified as a action verb. The existence of the case roles *agent* (`Peter`), *source* (`von Dudweiler`), and *destination* (`nach Saarbrücken`) leads to the proposal that `gruspel` is a subconcept of the FSS concept *motion*. Possible further results depend on whether one decides upon an interactive classification [FS84] and thus, strive for a refined concept, or, by comparing with similar concepts within the semantical lexicon, one finds significant criteria allowing for an interactive completion of the lexicon entry. Representative for this aim of concretion is the IRACQ system described by Weischedel et al. [WAH*87].

4.3 WASTL and KADS

With the help of the KADS methodology, it should be possible to formally specify the knowledge acquisition methods described above. The choice of a common terminology for describing both the knowledge acquisition procedures and the application system allows them to be represented in a single interpretation model. WASTL's inference structures are interpreted by means of inherent task structures. Knowledge on the strategic level is responsible for selecting these task structures.

The following examples shows how the construction of morphological lexicon entries, introduced informally in the previous section, can be both, formally described and combined with the normal morphological analysis by using KADS.

4.3.1 Inference Level

Figure 5 shows the inference structure of morphological analysis during its normal mode of operation and during knowledge acquisition. Two inference lines (labeled 1 and 2) can be seen. Inference line 1 refers to possible inferences in the normal operational mode. The central procedure is the word-for-word look up (`lookup`) within the morphological lexicon for the construction of the preterminal chain (`preterminals`).

As an alternative, inference line 2 has also to be considered. Here, a heuristic classification occurs as defined in [BWvS*86] (`observables` → `[transform]` → ... → `[specify]` → `solutions`). Intermediate results are morphological analysis hypotheses (`hypotheses-1`) which, in place of reliable, lexicon-based morphological results, are used for the

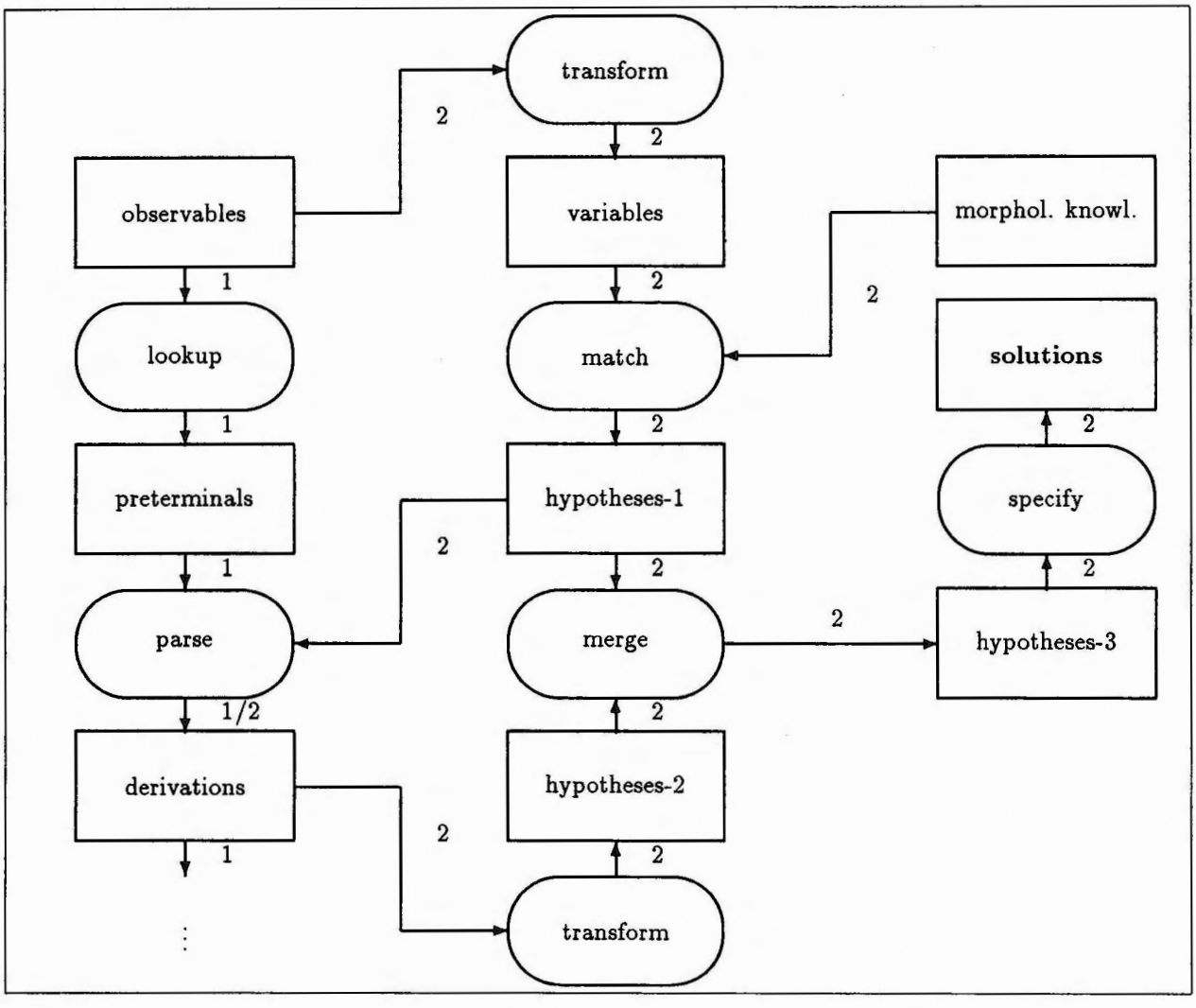


Figure 5: Morphological Analysis with and without Knowledge Acquisition

syntactical analysis (`parse`). The results of syntactical analysis (`derivations`) are examined as to which morphological analysis is allowed for a syntactically well-formed derivation. Combining the original hypotheses (`hypotheses-1`) with the results obtained in this second step (`hypotheses-2`) should reduce the search space (`hypotheses-3`) and allow for lexicon entry suggestions (`solutions`).

4.3.2 Task Level

Two tasks are responsible for the procedural interpretation of the inference structure: **simple-morphology** and **morphology-with-acquisition**. Basically, the tasks sequentially evaluate the inference structure:

task simple-morphology :
    LOOP for every observable word
        lookup(observable/word) **in** (morphological lexicon)
    ENDLOOP
    parse(preterminal chain)

task morphology-with-acquisition :

LOOP for observable word(s)

transform(observable/word)

[A] **match**(variables) **using** (morphological knowledge)

ENDLOOP

parse(data/preterminal chain)

LOOP for every derivation

LOOP for every word from [A]

[B] **transform**(derivation) **into** (hypotheses)

ENDLOOP

ENDLOOP

merge(hypotheses) **from** [A] and [B]

specify(solution/lexical entry)

4.3.3 Strategic Level

The decision as to which task should carry out morphological analysis occurs on the strategic level and is determined by various factors. Usually, the task **simple-morphology** is applied to each word of the input text. If the **lookup**-inference for a particular word should fail, the decision would be made as to whether the task **morphology-with-acquisition** should be activated. Parameters that might influence this decision are the operational mode of XTRA (is knowledge acquisition desirable or impossible), or, the chance of success of this strategy [5].

4.4 Applications of this Approach

The method described above also allows a step-by-step integration of further knowledge acquisition methods into the overall system. Concrete concepts for the expansion of the semantical lexicon already exist [6]. The domain-dependent knowledge bases, the conceptual knowledge base (CKB) as well as the transformation rules of the FSS onto the CKB and vice versa, should be able to be expanded in a similar manner.

This represents a mere first-step as to what can be accomplished on the basis of the previous discussion. In the future, it might be to disclose contradictory and incomplete data within the domain dependent knowledge base. Contradictions arise when hypothesis found during the knowledge acquisition process do not correspond with the intensions of the user. Answers to questions like "`Why not ...`" should be made possible on the basis of specification, knowledge concerning functional relationships, and on the activation of single application models.

Referring to the example in which no morphological entry for the verb existed, the user would oppose the suggested hypothesis that, a verb be entered into the lexicon. This he could express with the question "`Why is gruspel not an adjective?`". A rational answer requires knowledge concerning the constructing of the respective hypothesis. A possible answer might be that in the application of morphological knowledge, the word category adjective had been considered. On the other hand, the grammar, with its word

[5] A reduction of search space would become less productive the more undefined words contained in the input because, in this case, the selective effect of syntactical analysis diminishes.

[6] A detailed description will be printed at a later time.

category hypothesis adjective, has considered the input syntactically ill-formed and impossible to derivate. Should `gruspel` actually be an adjective, it would be necessary to enter appropriate grammar rules into XTRAGRAMM.

The techniques in achieving this are known from explanation components of expert systems since [Dav82] and from examining algorithmic debugging procedures within AI-programming languages [Sha83,SL86].

The natural language description of knowledge base contents can also help further expand and develop knowledge bases. Within XTRA, the necessary tool – the generation component – exists and is a central part of the application system. One possibility of utilizing the NL-description of knowledge base contents might be the technique of paraphrasing, which, for user inputs, basically serves to disclose ambiguities or misinterpretations. This is achieved by interrupting analysis of the user input up to a desired point (e.g. the sentential semantic analysis producing FSS individualizations), marking the attained analysis results on this level as being relevant for the output, and finally, generating the natural language output.

Again, this procedure can be described through a task operating on the inference structure. A task structure for paraphrasing FSS indivualizations might be as follows:

task paraphrase-fss:
    simple-morphology(observable/text)
    parse(preterminals)
    sentential-semantic-analysis(derivations)

    mark(fss)
    generate(fss)

Desirable is not only the sufficient access to explicit representation of the interpretation model and the availability of the tasks needed to date but also, the possibility of an interactive description and interpretation of new tasks.

5 Summary and Perspectives

In this paper, two concepts for knowledge acquisition were introduced and applied to a complex knowledge based natural language system: the example-driven knowledge acquisition and the integrated specification of application and knowledge acquisition system. Several components of the WASTL system are the subject of detailed investigations in the framework of master theses, including the acquisition of morphological and semantical lexicon entries as well as an environment for interactively describing and interpreting new inference and task structures.

The objective of the studies will be to extend WASTL to a workbench which can be used to experiment with different knowledge acquisition and machine learning approaches. XTRA is a sufficiently complex application to investigate whether or not these approaches are suitable for real world problems. The integration of the approaches into WASTL and XTRA on the one hand and an improved comparability on the other hand is achived through the re-specification with the help of the KADS methodology.

WASTL and XTRA are being developed within the framework of the German Special Collaborative Program on AI and Knowledge Based Systems (SFB 314) of the German Science Foundation.

References

[AB86] Glenn Abrett and Mark H. Burstein. The KREME Knowledge Editing Environment. In *Proceedings of Knowledge Acquisition for Knowledge Based Systems Workshop*, 1986.

[AHK*88] J. Allgayer, K. Harbusch, A. Kobsa, C. Reddig, N. Reithinger, and D. Schmauks. *XTRA: A Natural Language Access System to Expert Systems.* Report, Dept. of Computer Science, Universität des Saarlandes, Im Stadtwald, 6600 Saarbrücken, West Germany, 1988. To Appear.

[Bal86] Bruce W. Ballard. User Specification of Syntactic Case Frames in TELI, a Transportable Natural Language Processor. In *COLING 1986*, pages 454 – 460, Bonn, 1986.

[BP85] Henning Bergmann and Annedore Paeseler. *Entwurf und Implementation einer Wissensakquisitionskomponente für ein natürlichsprachliches Datenbankzugangssystem.* Technical Report FBI-HH-B-117-85, Universität Hamburg, Fachbereich Informatik, 1985.

[Bra82] R. J. Brachman. On the Epistemological Status of Semantic Networks. In *Readings in AI and Knowledge Representation*, Morgan Kaufmann Publisher, 1982.

[BS86] B. W. Ballard and Stumberger. Semantic Acquisition in TELI: a Transportable User-Customized Natural Language Processor. In *Proceedings of the 24rd Annual Meeting of the ACL*, Columbia University, New York, 1986.

[BWvS*86] J. Breuker, B. Wielinga, M. van Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, J.-P. Billault, M. Davoodi, and S. Hayward. *Model-Driven Knowledge Acquisition: Interpretation Models.* Memo 87, Departement of Social Science Informatics, University of Amsterdam, Herengracht 196, 1016 BS Amsterdam, The Netherlands, 1986. Esprit Project 1098.

[Dav82] R. Davis. TEIRESIAS: Applications of Meta-Level Knowledge. In R. Davis and D. B. Lenat, editors, *Knowledge-Based Expert Systems in Artificial Intelligence*, McGraw-Hill, 1982.

[FS84] Tim Finin and David Silverman. *Interactive Classification: A Technique for the Acquisition and Maintenance of Knowledge Bases.* Report MS-CIS-84-17, University of Pennsylvania, COmputer and Information Science, 1984.

[GAMP85] B. Grosz, D. E. Appelt, P. Martin, and F. Pereira. *TEAM: An Experiment in the Design of Transportable Natural Language Interfaces.* Technical Report 356, SRI International and Center for the Study of Language and Information, Stanford, 1985.

[Jan87] Roman M. Jansen-Winkeln. LEGAS: Inductive Learning of Grammatical Structures. In *Proceedings of the AISB '87*, John Wiley Ltd, Edinburgh, 1987.

[Jan88] Roman M. Jansen-Winkeln. Spezifikation of a Knowledge-Based Natural Language System. 1988. Submitted at GWAI 1988.

[Kar86] L. Karttunen. *D-PATR: A Development Environment for Unification-Based Grammars.* Working Paper, SRI International and Center for the Study of Language and Information, Stanford, 1986.

[KB82] Ronald M. Kaplan and Joan Bresnan. *Lexical-Functional Grammar: A Formal System for Grammatical Representation*, chapter 4. The MIT Press, 1982.

[Mos83] M. Moser. *An Overview of NIKL.* Technical Report 5421, BBN Laboratories Incorporated, Cambridge, MA, 1983.

[PB85] S. F. Pilato and R. C. Berwick. Reversible Automata and Induction of the English Auxiliary System. In *Proceedings of the 23rd Annual Meeting of the Association of Computational Linguistics*, Chicago, 1985.

[PB87] J. Pustejovsky and S. Bergler. On the Acquisition of the Conceptual Lexicon. 1987. submitted to AAAI-87.

[Sch88] Klaus Schifferer. TAGDevEnv: Eine Werkbank für Tree Adjoinig Grammars (TAGs). In *Tagungsband: Theoretische Grundlagen der Computerlinguistik*, Springer Verlag, 1988. To appear.

[Sha83] E. Shapiro. *Algoritmic Program Debugging.* MIT Press, Cambridge, MA., 1983.

[SL86] Leon Sterling and Marucha Lalee. An Explanation Shell for Expert Systems. *Computational Intelligence*, 2:136–141, 1986.

[Wah82] W. Wahlster. Natürlichsprachliche Systeme. In *Künstliche Intelligenz*, Frühjahrsschule Teisendorf, Springer Verlag, 1982.

[WAH*87] R. Weischedel, D. Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked, and D. Stallard. *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program.* Annual Technical Report 6522, BBN Laboratories Incorporated, Cambridge, MA, 1987.

[Wot84] Klaus C. Wothke. *Maschinelle Erlernung und Simulation morphologischer Ableitungsregeln.* PhD thesis, Universität Bonn, 1984.

KNOWLEDGE ELICITATATION FOR SECOND GENERATION EXPERT SYSTEMS

ABSTRACT

AUTHORS: N.E. JOHNSON C.M.TOMLINSON

Department of Computer Science
Brunel University
Uxbridge, UB8 3PH
U.K.
E-Mail nancy@cs.Brunel.ac.uk

0. INTRODUCTION

In this paper we will outline the requirements of techniques for eliciting and analysing knowledge to be used in second generation expert systems. Most of these needs are global and are of just as much concern for developing KBS.
The term First Generation refers to systems for which there is a rule based architecture, often but not exclusively typified by the use of a shell. A Second Generation system is here taken to be using an architecture with some explicit representation of reasoning or strategy (e.g. Generic Task Structures from Chandrasekran, Competence Models from Johnson, L.). A correspondingly more sophisticated system development environment (e.g. a knowledge engineering environment) is commonly used.
We are concerned with addressing the problem of how well knowledge elicitation techniques can scale up to meet the demands of building the more structurally complex second generation systems for it seems that the computer power is available but knowledge engineering methods are still primitive.
We will argue (next section) that the available automated knowledge acquisition tools, including sophisticated knowledge base editors, do indeed support the 'knowledge book-keeping' tasks but do not support the conceptually difficult tasks of early conceptualization of the domain nor the later stage of validating the knowledge. Both are essential. The first because early elicitation is 'creative' and so can easily degenerate into an undisciplined selection of 'knowledge' depending on that which the knowledge engineer understands or that which the system can accept. The second is important because good practice in KBS and expert systems development demands that claims made for the validity of the knowledge in the system be tested against a wider population than the original expert. To this end we suggest the use of 'Mediating Representations' (Johnson NE, 1987b) as a good tool to help proceed in a disciplined yet flexible manner.
The paper will describe the demands made on Mediating Representations (Section 3) with examples (Section 4&5) drawn from current work in ESPRIT 1592 A Therapy Advisor for Oncology (a debut performance!)

1. TOOLS AVAILABLE FOR ELICITATION: scaling up!

(We include an extract from Johnson 87b to show the flavour of the arguments and authors' ability to write prose!)

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 using Interview techniques is widely acknowledged to be valuable. More rarely is it admitted that 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. For a more complete discussion of the notion of a Mediating Representation see Johnson, NE 1987b).

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 requirements analysis (= elicitation) will pay dividends in time and effort during the later stages of prototype implementation.

(extract ends)

CURRENT TOOLS

(This section will contain remarks on how current tools support early and later stages of knowledge elicitation, for example,

AQUINAS type tools work most well after the expert accepts the domain can be divided into constructs.

KADS works best if the interpretation model has alraeady been appropriately selected and if the analysts accept Brachman's computer based model as an appropriate model for human expertise.

Many systems are very efficient tools for 'cleaning up' the knowledge after the initial conceptualization has been made (e.g. ASTEK, KNACK, MORE/MOLE, ASK).

We wish to develop methods and tools to make explicit and support the early selection process.
Also, given our style of interviewing that aims to make efficient use of the expert's time (i.e short sessions on site) we need a representational form which is more 'expert friendly' than AI knowledge representation languages and which does not rely on the use of high powered work stations to do even the simplest task of displaying summaries of structured knowledge.

A warning - that big trees on screens with superb editing facilities does not constitute knowledge analysis. Examples drawn from current tools showing that knowledge base editors can give rise to an illusion that the complexity of knowledge can be represented by adding yet another node or link. Also that the expressiveness of the links is poor in that many distinctions in the domain can be masked by the use of associative links in networks. (At this point we are doing no more than recapitulating the argument from Clancey about compiled knowledge in Mycin rules).

So, if we accept that some more structuring tools are required, the question is, what sort of tools. We suggest tools which help the analyst to construct a Grammar of the expert's knowledge.

3. KNOWLEDGE STRUCTURING AND MEDIATING REPRESENTATIONS

THE COMPUTATIONAL MODEL METAPHOR AND THE GRAMMATICAL MODEL METAPHOR

The Computational Model is the default model in Knowledge Engineering. In this section we will summarise the effects on knowledge elicitation, many having already been mentioned in previous text.
Then the Grammatical Model will be advanced as an alternative, and more appropriate approach to knowledge elicitation, with outlines of requirements on knowledge elicitation tools in general (eg expressive formalisms, limitations on power of grammar...). One requirement is the Mediating Representation.

MEDIATING REPRESENTATIONS

What They Are!

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).

What They Are Not!

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 more than a version of the linguists', and latterly knowledge engineers', notion of 'intermediate representation'. An intermediate representation has a more temporary status. It is a transient, indeed intermediate, repository for the knowledge which becomes redundant once the knowledge has found expression in either another intermediate representation or in the final form of codable knowledge. We ask our mediating representation to be more in that it should hold a permanent record of the structure of the knowledge (the grammar) from which several selections made be made of implementable knowledge. It acts as a communication medium between all members of the knowledge engineering team at all stages of system building and evaluation. In addition we prefer the term Mediating Representation to convey the sense of synthesis and coming to understand through the representation which is constitutive of knowledge analysis.

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 (a generic example will be given).

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, as we argued in section , 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 section 1 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. In the real example of section 4 we show how this was 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. (cf early work with KADS).
Using the approach of Grounded Theory to generate names for distinctions in the domain and expertise and encapsulate those terms 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.

4. 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 we will give a simple example on a domain appropriate to a European audience (guaranteed not to be car ignition!).

5. A REAL EXAMPLE

This section will contain an example of an SGN from our current research in ESPRIT 1592 TAO (A Therapy Advisor for Oncology).
The example will be about how to decide whether a patient can receive a specific course of chemotherapy (inclusion in the protocol).
Comments will be makde on how this SGN was used

- as a knowledge elicitation tool
- as a medium of communication with the knowledge implementation team
- how the SGN relates to the system architecture in terms of task trees, intelligent data handling and strategic explanations
- plans for testing the validity of the knowledge with other experts.

6.Postcript

BIBLIOGRAPHY

Banff Conference 86
Banff Conference 87

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

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).

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

Johnson, N. (1987b) Mediating Representations in Knowledge Elicitation. EKAW87, Reading University, 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.

Incremental and Reversible Acquisition of Taxonomies

Jörg-Uwe Kietz

Technische Universität Berlin
Projektgruppe KIT
Sekr. FR 5-12
Franklinstr. 28/29
1000 Berlin 10
West Germany
E-mail: kietz%db0tui11.bitnet@cunyvm.cuny.edu

Abstract

One of the prerequisites for building up a domain model is the specification of a description language. For a logic-oriented representation, domain-specific predicates have to be declared along with their admissible arguments. If a many-sorted logic is used for representing the domain model, the sorts of the arguments, the compatibility of the sorts, and the mapping of terms to sorts have to be declared, as well. This specification can be viewed as a knowledge acquisition task itself.

In this paper, an implemented approach to the automatic acquisition of a sort taxonomy is presented. The sort taxonomy is used to specify the well-sorted expressions in a many-sorted logic, i.e. to specify the logic description language for a particular domain model.

The taxonomy is constructed on the basis of facts inputted by the user. It is assumed that the user inputs only semantically well-formed expressions, even though he is not forced to specify the description language beforehand. The criterion for their well-sortedness can thus be derived from the meaningful facts.

The taxonomy evolves incrementally as more and more facts are typed in. Revisions in the set of facts produces revisions in the taxonomy. If a new fact input is not well-sorted with respect to the taxonomy acquired so far (e.g. if the user has forgotten how he used a certain predicate before), this is indicated to the user. The system does not, however, prevent the user from revolving the taxonomy. In contrast, the resulting changes are computed by the system automatically.

1 Introduction

Many AI systems presuppose taxonomies which specify the terminology used to perfom their tasks. Natural Language systems (e.g. LIFER [Hen 77]) store the necessary conceptual knowledge to understand language in form of a taxonomy, many hybrid Knowledge Representation systems (e.g. KRYPTON [Bra 85]) define the terminology of the represented domain in taxonomies (e.g. in KL-ONE [BrS 85] like taxonomies), and several Machine Learning approaches (e.g. Structural Matching [KoG 86], LEX [MUB 83]) provide taxonomical background knowledge for generalization or spezialization of the learning results. Most often, the used taxonomies are hand-coded by the system designers or the knowledge engineers. If a Knowledge Acquisition

system (e.g. ONTOS [Mon 87]) is used, it often supports only the top-down refinement of the acquired taxonomy by asking the user for taxonomic relations. Knowledge Elicitation systems (e.g. AQUINAS [BoB 87]) construct taxonomies by asking the user for attributes and values of concepts and combine them by rating-grids. In these Knowledge Acquisition systems incremental acquisition is restricted to refinement, revisions are not supported, and revolving the taxonomy is hard to perform.

There are also Machine Learning approaches (e.g. Conceptual Clustering [MiS83], [MiS86]) for automatically generating taxonomies of objects or events. However, learning an object taxonomy itself presupposes a fully specified representation language, including local taxonomies specifying the given argument-types. The representation language and the local taxonomies for the arguments must be hand-coded in advance by the user, and mistakes in this declaration result in useless object taxonomies.

In this paper, a fully implemented approach for the automatic acquisition of sort taxonomies is presented. In contrast to knowledge elicitation systems which query the user in order to build up the taxonomy, this acquisition is no distinct step, but is integrated into the acquisition of factual knowledge. This means that the user is not asked for taxonomic knowledge, but that the sort information of the user's input is evaluated to build the taxonomy. The approach is based on the assumption that the user always inputs semantically well-formed expressions - the user can't even think of senseless (as opposed to false) expressions. This assumption allows constructing the sort taxonomy and the mapping of terms into sorts by exploiting the fact inputs of the user. Therefore, the acquisition of a taxonomy is fully integrated as a side-effect of the overall acquisition procedure.

In contrast to Conceptual Clustering, only the relations between sorts are computed and no intensional characterisations of the sorts are generated. Thus, the acquired taxonomy has no predictive power, but is a description of the language used by the user to input facts. In other words a specification of the description language is not presupposed, but produced.

The user input is used to incrementally build up the taxonomy, which therefore can be presented to the user at any time during construction. During construction, the system may also work upon the taxonomy representing the current state of modeling. This process is not only incremental, but reversible, so that the user need not be concerned with the changes of the taxonomy implied by changes of the domain model. The taxonomy construction is achieved by structuring the known term sets for each argument in the given predicates according to set theory, i.e. we take the set theoretical relations between the extensions of sorts for determining the relations between sorts. Therefore, an inductive step would be necessary to use the taxonomy for predictions. So far, this step has not preformed by our system in order to preserve the incrementality and reversibility of the approach.

In section 2, we describe the way we get from a set of facts to a sort taxonomy. The terms are mapped into sorts corresponding to the argumentplaces of the used predicates. For every argumentplace of every predicate, a unique argsort is generated. This can be done automatically by the system without making a priori predicate declarations. Using this non-intuitive definition of sorts, we can compute an adequate representation of sorts by dividing the set of sorts into equivalence classes of sorts. These classes represent what is normally called a sort: a definite name for argumentplaces with the same extension. These classes are arranged along a partial order representing the compatibility-relation between sorts. The partial order is derived from the subset-relation between the extensions of the sorts. Finally, we determine the overlappings between sorts and reprensent them in the taxonomy to distinguish between overlapping and disjoint sorts. By doing this, we get a lattice of equivalence classes of sorts.

In section 3, we present transformations of the sort lattice which make an incremental and reversible acquisition of the sort lattice possible. These transformations describe the changes of the sort lattice corresponding to changes of the set of facts. This enables the incremental acquisition of the sort lattice by adding new facts, beginning with no facts and the empty lattice.

In section 4, we introduce some possible applications of our approach. Finally, we describe an implemented application within the Knowledge Acquisition system BLIP. The sort lattice is used to specify the description language of the domain models acquired with BLIP.

2 The Sort Taxonomy

In this section, we present the taxonomy of sorts computed from a set of known facts. First we define the correspondence between argumentplaces and sorts. With this correspondence, we can compute the extensions of the sorts, i.e the sets of terms belongs to the sorts. Using the extensions of the sorts, we can compute the the most important relations between sorts:

- the equivalence of sorts,
- the inclusion of sorts,
- the overlapping of sorts, and
- the disjointness of sorts.

It can be shown that a sort lattice representing these relations can be constructed from the sort information residing within the set of facts. From this lattice, a bijective extension mapping to a complete infimum-homomorph lattice of the lattice of the power set of all terms (Pot(TERM)) exists. This mapping transports into the sort lattice part of the valid laws of the boolean algebra

(Pot(TERM), $\cap$, $\cup$, $\neg$, {}, TERM) with the partial ordering $\supseteq$,

being expected intuitively. The partial ordering (subset relation) and the infimum operation (intersection) may be transported from the extensions into the sort lattice. We can define the supremum operation so that it is as similar as possible to the union of sets.

2.1 The Mapping of Terms into Sorts

In a many-sorted logic, all predicates must be declared before using them. In predicate declarations for each argument, a sort to which the used arguments must belong is specified. So as not to trouble the user, the system itself generates the declarations of predicates when a predicate is used the first time. Normally, arguments with equal sorts are declared with identical sortnames, but our system defines a unique sort for each argument of each predicate and automatically calculates the equality of argument sorts within the sort taxonomy by building equivalence classes of sorts.This prevents several problems. It is possible to declare predicates automatically with shared sortnames on the first occurance of the predicate only if all given terms have already been mapped to sortnames. Still, problems may arise because of ambiguous mapping to multiple sorts. In our approach, a new unique sort is invented as a mutual sub-sort of the sorts containing the term; this new sort automatically climbs to a more adequate position within the sort lattice as the set of known terms for this sort grows. This method produces the finest and most appropiate sort taxonomy based on the current set of facts. For this reason, the set of well-sorted formulas is restricted as much as possible to formulas supported by facts, e.g. two argument sorts are equal only if they are mapped to the same extension. Finally, the system generates no a priori declarations of predicates which may prove inadequate and must be revised.

By collecting all terms used in facts for one argumentplace of a predicate, we derive the set of terms belonging to the argument sort (the extension of the argument sort). For each argument sort generated for an argumentplace of a predicate, an extension can be derived by doing this collection. These extensions are the base for calculating the relations between the argument sorts.

2.2 The Equivalence Classes of Argument Sorts

In our approach the most important relation between argument sorts is their equality. This relation is handled by the decomposition of the set of argument sorts into equivalence classes of argument sorts. Argument sorts are mapped into the same equivalence class if they have the same extension. This causes an injektiv mapping between the equivalence classes and the extensions. From now on, we will speak about the extensions of classes, meaning the equal extension of the contained argumentsorts. The classes represent what is normally named a sort: the equality of arguments. The computation of equivalence classes shows which argument sorts have an equal extension.

2.3 The Compatibility Partial-Order between Classes

After determining which argument sorts are equivalent, we must determine which argument sorts are compatible. The compatibility is used to decide the sort-correctness of variable bindings in formulas.The compatibility relation is computed between the classes because they are representative and definite. A compatibility between classes induces a compatibility between the ar-

gument sorts contained in the classes. As a minimal requirement, this relation must be a partial-order ([Obe62]). In our case, the compatibility partial-order is determined by the subset partial-order between the extensions of the classes. A class c_1 is a subclass of a class c_2 iff the extension of c_1 is a subset of the extension of c_2, i.e. a class c_1 is a subclass of c_2 if all terms belong to c_1 also belongs to c_2. This corresponds with the usual understanding of subclass.

2.4 Intersection sorts

Often, there are overlaps between two or more classes without any one class being a subclass of any of the others. To distinguish these overlapping classes from disjoint classes, we need a mutual subclass of the overlapping classes. An example of two overlapping classes would be the classes containing 'liquid' and 'drug'. A mutual subclass of these classes would be the class containing 'serum'. To represent the relation between these classes within our sort lattice, the system generates additional sorts representing the overlaps between classes. For this purpose, we define intersection sorts as the named intersection of classes extensions. Therefore, intersection sorts are artificial and represent not arguments of predicates, but possible sorts of variables. We are thus able to distinguish between disjoint and overlapping classes. Disjoint classes have an infimum equal to NIL, while overlapping classes have an infimum greater than NIL.

By completely generating all intersection sorts, we finally get a complete infimum-homomorph lattice of Pot(TERM), i.e. the intersection of several sorts is always defined by the infimum of their classes within the sort-lattice.

3 The Acquisition of the Sort Taxonomy

In order to make the acquisition of the sort lattice incremental and reversible, we have to define operations transforming a sort lattice according to factual changes, i.e. transforming the sort lattice from its old state into a new one corresponding to the new factual knowledge. Starting with an empty lattice, the lattice corresponding to the current state of modeling can be constructed by means of the following operations.

On the one hand, the sort lattice must be represented as completely as possible in order to keep the expense of computing important information for the system and the user low. On the other hand, the representation should be highly modular to keep the expense of the transformations low. One solution is to represent the partial ordering and intersection sorts between equivalence class names instead of argument sort names, which yields a highly modular representation. If we hold the mapping between class names and extensions constant, only the membership of argument sorts to classes changes if the argument sort's extensions change, while the existing partial ordering and the existing intersection sorts remain unchanged. All we have to do is to compute a new class for the changed sort and possibly purge the old class of the argument sort.

If a changed sort fits into an existing class, we must only add the sort to the class. If the changed argument sort does not fit into any existing class, a new class with a new class name has to be created. The position of the new class within the partial ordering has to be determined, and all of its intersection sorts have to be built and integrated into the class hierarchy. The position in the partial-order can be determined from the position of the old class of the changed argument sort. Intersection sorts with the new class can also be built and integrated into the lattice using the intersection sorts of the old class. Integration of the new intersection sorts can be done by applying this method recursively.

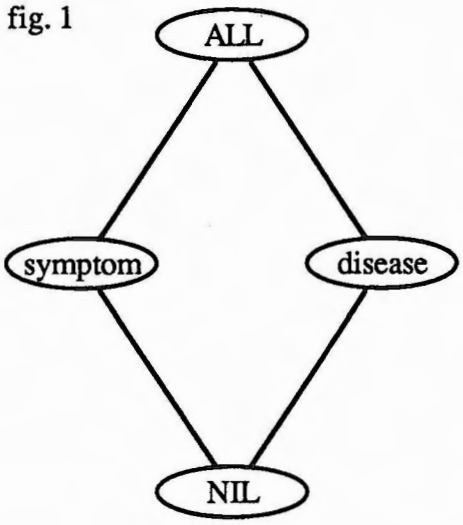
The old class must be deleted if it contains no sorts after integrating the changed argument sort and the new intersection sorts. When a class is deleted, all of its intersection sorts and its position within the partial ordering have to be deleted. If this produces more empty classes, they must also deleted.

The average expense of this approach is lower than computing the relations directly between the sorts since it is easier to determine what remains unchanged and what has to be changed.

3.1 An Example

To illustrate the acquisition of a sort taxonomy we would like to give an example from the 'Side-Effects of Drugs' domain. In fig. 1 - 4 & 6, only the classnames the user has in mind (the system generate only classnames like 'class_1', but supports the naming of classes by the user) are represented by nodes and the subclass relation between classes is represented by links. Only in fig. 5 are the equivalence classes of sorts themselves and their known extensions shown.

fig. 1



We would like to start the domain modeling with the relations between symptom and disease. These relations can be expressed by the two predicates, 'indicate' and 'cause'. By inputting the facts

indicate(sore_throat,flu) and cause(flu,sore_throat)

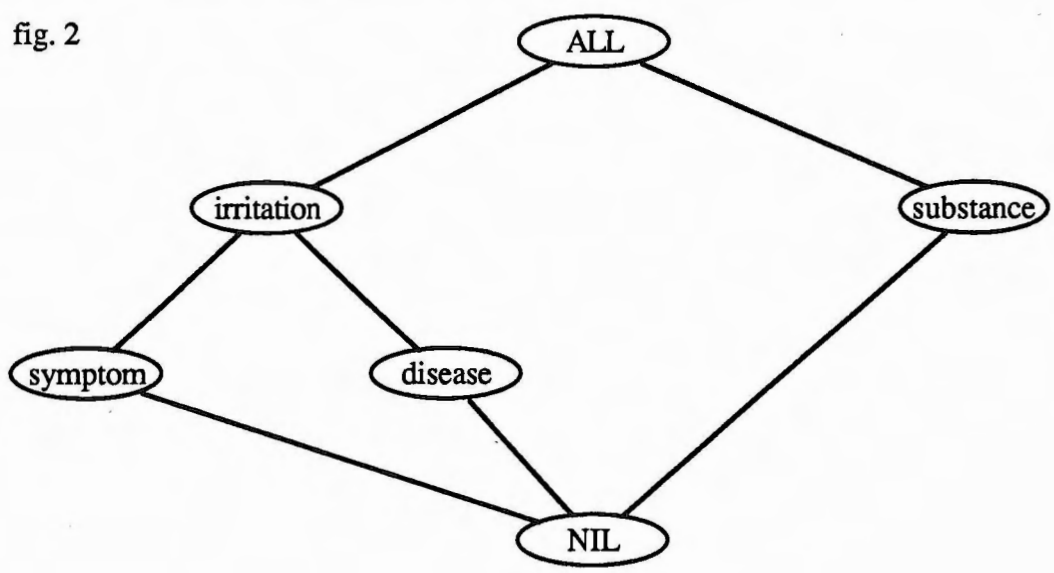
into the system, it creates the sort taxonomy shown in fig.1. Here, nodes represent classnames and links represent subclass relations between classes. The classname 'symptom' represents the equivalenceclass [arg_1_indicate, arg_2_cause] and the classname 'disease' represents the equivalenceclass [arg_2_indicate, arg_1_cause]. The system has thus established the equivalence of the argument sorts of the first argument of the predicate 'indicate' and the second argument of 'cause' as well as the equivalence of the argument sorts of the second argument of 'indicate' and the first argument of 'cause'. The known extension of class 'symptom' is {sore_throat}. The known extension of class 'disease' is {flu}. Therefore it establishes the two classes be disjoint, i.e. the system does not know any term which belongs to the extensions of both classes.

The system so far knows only 'ALL' as a mutual superclass. This is changed when the system receives facts about positive effects of substances, like

affects_pos(inspirol,sore_throat), affect_pos(aspirin,flu),
affect_pos(bc, sore_throat), affect_pos(asa,flu).

The system builds two new equivalenceclasses containing the argument sorts of the 1st and the 2nd argument of the predicate 'affect_pos'. They become new classnames. In fig.2, they are named 'substance', containing the argument sort of the 1st argument of 'affects_pos', and 'irritation', containing the argument sort of the 2nd argument. The system finds out that the extension of 'symptom' as well as the extension of 'disease' are subsets of the extension of 'irritation', i.e. that substances affect diseases as well as symptoms. Therefore, it not only creates 'substance' as a new equivalence class disjoint to all other classes within the taxonomy, but also 'irritation' as a generalization of 'symptom' and 'disease'.

fig. 2

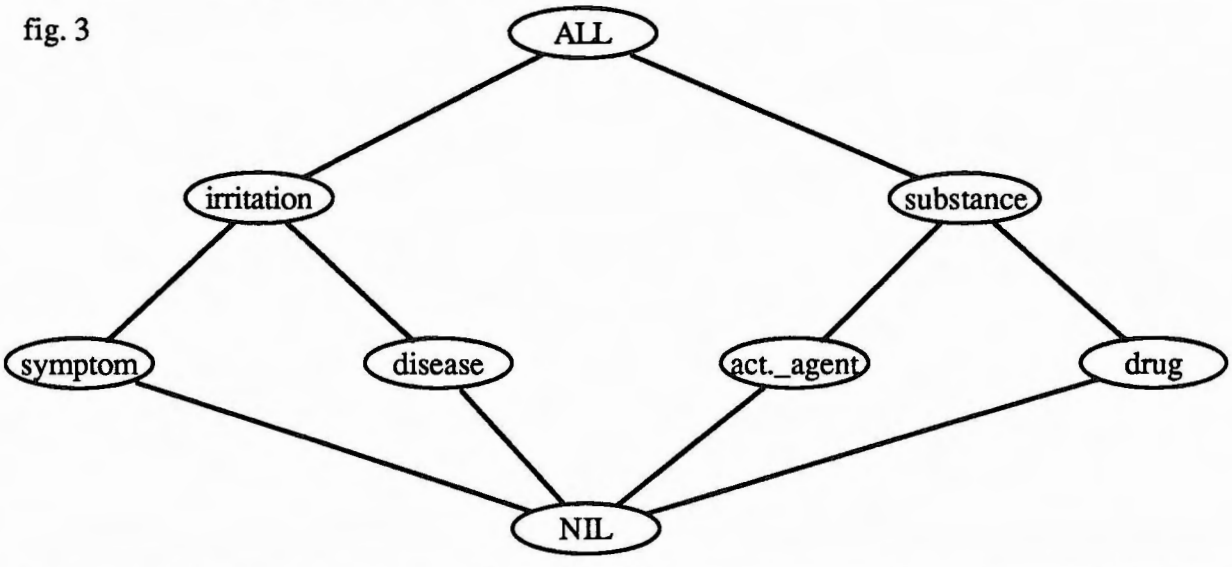


Specialization of equivalence classes is also performed by the system. The facts

contains(inspirol,bc), contains(aspirin, asa)

lead to the creation of the equivalence classes with the names 'act._agent' (with the argument sort arg_2_contains) and 'drug' (with the argument sort arg_1_contains) as specializations of 'substance'. The resulting taxonomy is shown in fig. 3.

fig. 3

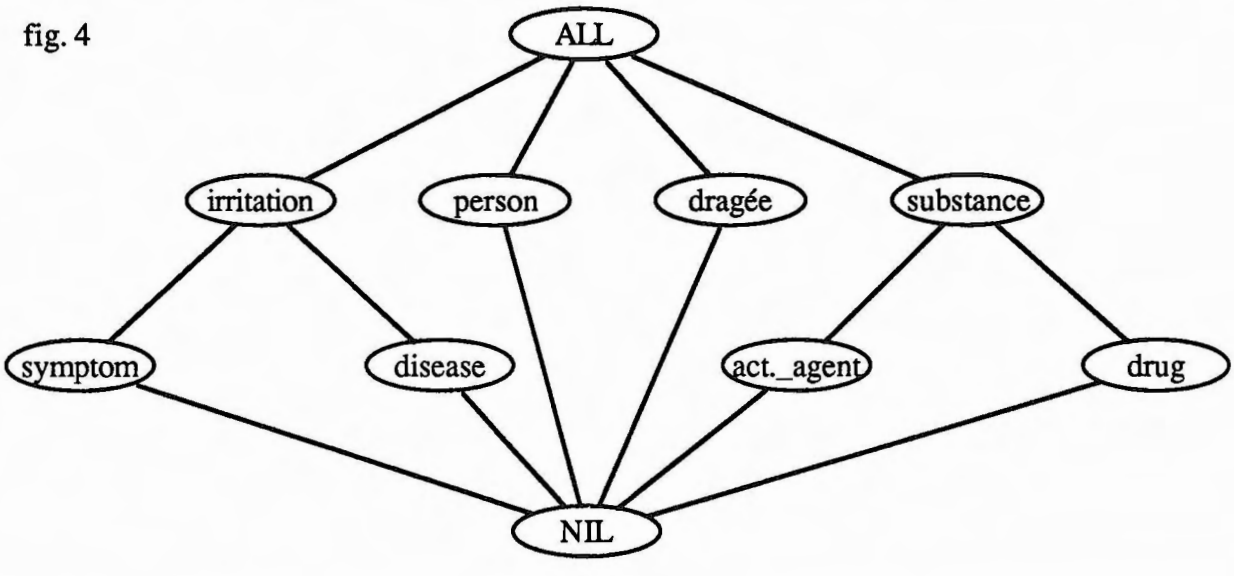


Two new equivalence classes named 'person' (with the argument sort 'arg_1_suck') and 'dragee' (with the argument sort 'arg_2_suck') are introduced by giving the fact

suck(uwe,vivil).

They are completely disjoint from all other existing equivalence classes as shown in fig. 4.

fig. 4

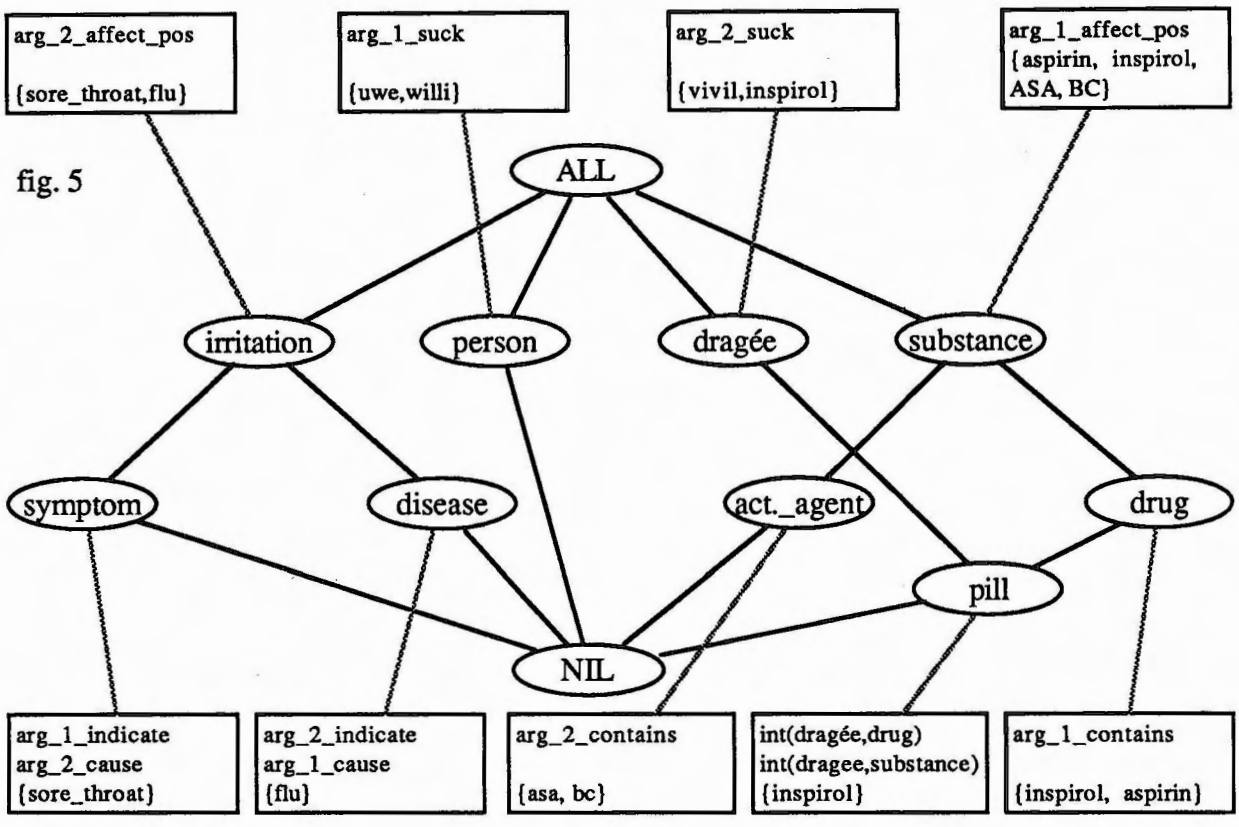


If we input the fact

suck(willi,inspirol)

the system will diagnose overlaps between 'dragée' and 'drug' and between 'dragee' and 'substance',both consisting of the term 'inspirol'. For each overlap, an intersection sort representing this overlap is built. Both intersection sorts (int(dragée, drug) and int(dragée, substance)) have the extension {inspirol}.They are integrated into the taxonomy by building the new equivalence class named 'pill' as a mutual subclass of 'dragée' and 'drug'. The resulting taxonomy is shown in fig. 5. In this figure, all equivalence classes are shown precisely as represented within the system. For each class, the contained argument sorts and the current extension are shown.

fig. 5

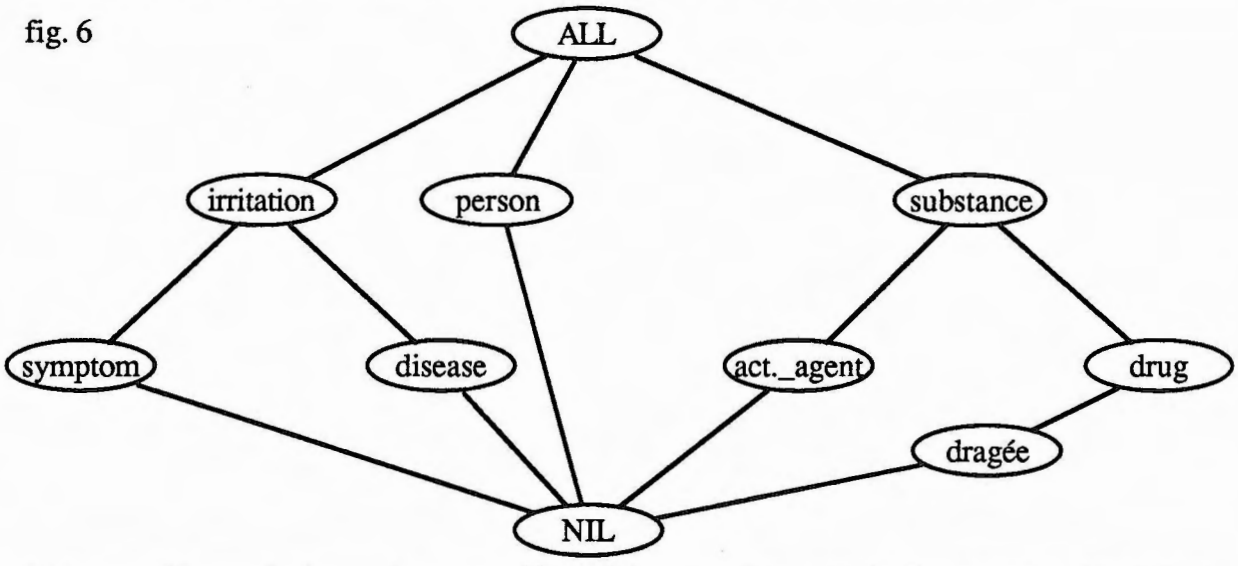


The created taxonomy is independent of input sequence; the facts of our example may be given in any order. To demonstrate this, we swap the order of input for the last two facts. If

suck(willi,inspirol)

were input in the situation of fig. 3, the taxonomy of fig. 6 would be the result.

fig. 6



All terms of the equivalence class named 'dragée' are now known to be drugs, so the class is integrated into the taxonomy as a specialization of 'drug'. For the input

suck(uwe,vivil)

this condition is no longer true. 'dragée' is no longer a specialization of 'drug'. For this reason, 'dragée' becomes a new equivalence class at the level of 'substance'. The old class now precisely describes the intersection sorts between 'drug' and 'dragée' and 'substance' and 'dragée'. If it is named 'pill', the taxonomy of fig. 5 is reached again.

The changes in the taxonomy caused by the revision of facts are computed similar. The result of revisions is independent of sequence, too.

4 Applications

As an example of an environment using this approach and the incrementally aquired taxonomies, we would like to introduce its application within the Knowledge Acquisition system BLIP. In BLIP, a many-sorted logical language is used to represent the acquired domain models. The taxonomy is used to specify the well-sorted expressions of the language for the model acquired so far.

In general, the acquired taxonomy can be used to specify the well-sorted expressions of a many-sorted language. Of course, applications within other environments are possible, e.g. to acquire object taxonomies in expert systems or to acquire the taxonomies used to generalize or specialize the learning result in Machine Learning approaches like Structural Matching or LEX. The acquisition of the attribute-taxonomies presupposed by Conceptual Clustering to construct the object-taxonomy also seems possible.

Another application could be to translate the acquired sort lattice into a primitive KL-ONE taxonomy. A sort lattice may be considered as a simplified form of a KL-ONE taxonomy. The sort lattice also represents the domain taxonomy and is used in the predicate declarations to define the used terminology. Comparing a sort lattice with a KL-ONE concept taxonomy shows that both are lattices based on the inclusion partial ordering, with the sorts of the sort lattice corresponding to the primitive concepts in KL-ONE. For both, no intensional characterizations are available, i.e. for every sort or primitive concept all constants belonging to them must explicit given. The possible role fillers of KL-ONE roles are restricted by domain and range definitions with concepts. This corresponds to the restriction of possible arguments for the argumentplaces of the predicates through predicate declarations with sorts. However, KL-ONE's expressive power, with its number and value restrictions, the possibility of defining concepts and the role hierachy, is much greater than that of a sort lattice. Also, classification of concept desciptions has no equivalent in the sort lattice.

4.1 The Knowledge-Aquisition-System BLIP

The BLIP system ([Mor 87]) was designed with the aim of acquiring domain models. The BLIP user (expert or knowledge engineer) starts creating the model by giving known facts and rules of a domain to the system. In the 'sloppy modeling' paradigm [Mor 88], the system uses this input and system resident (and domain independent) meta-knowledge[Thi 88] to support further modeling by inferring the consequences of the modeling so far.

The system also uses a Model-Driven Machine Learning approch ([Wro 88]) to learn rules about the domain, i.e. to create and integrate rule hypotheses into the model. The system detects contradictions within the model and uses them to refine the model, thereby resolving the contradictions.

4.2 Acquisition of Sort Taxonomies in BLIP

In the knowledge acquisition system BLIP, the representation language for domain models is a many-sorted logic. The sort taxonomy and the predicate declarations are not presupposed, but are acquired by the above described approach by BLIP itself. Since the user of BLIP does not need to have an explicit model of the domain in mind when starting the modeling, but develops it only while interacting with the system, the many-sorted logic appropriate for the domain model cannot be construed beforehand, but must be aquired completely integrated into the modeling process. The acquisition of the sort taxonomy and the predicate declarations thus must be as incremental and reversible as is the acquisition of the overall model in order to describe the in the current state of modeling used language precisely.

Model construction, i.e. the model's permanent change and refinement, is performed in BLIP from extension to intension. This direction of modeling in BLIP is kept for the acquisition of the sort taxonomy.

When the user inputs a new fact into the system, this not only means that the fact is correct, but also implies that the mapping from terms to sorts is correct for this fact. This mapping can thereby be derived from factual input. As a result, when a fact is deleted, the mapping has to be reversed and the terms have to be deleted from the corresponding sorts.

Since the actual extensions are constantly calculated, the sort taxonomy precisely corresponds to the system's current state of factual knowledge.

The sort taxonomy is used to
- present the user an overview of the used predicates and the inputted facts,
- decide about the compatibility of the used predicates' arguments,
- survey correct usage of predicates,
- restrict the set of hypotheses to the verifiable ones, and
- compute the validity of taxonomic rules.

The sort taxonomy's graphic representation which corresponds to the current state of modeling is made available to the user, who thereby gets an overview of the domain structure, the known facts, and the used predicates.

Relations between different predicates' arguments are represented by the taxonomy. During the creation of the taxonomy, it is checked whether

- the argument sorts are equivalent, i.e. if they belong to the same equivalence class
- an argument sort is a subsort of another, i.e. if there is a subclass relation between their equivalence classes
- an argument sort is compatible with another, i.e. if there is a mutal subsort different than NIL
- argument sorts are incompatible (disjoint), i.e. their only mutual subsort is NIL

This specification of arguments allows checking the well-sortedness of factual and rule input. By this means, the system helps to prevent input errors, like accidentally swapping arguments of a predicate. If in the situation of fig. 5 the user inputs the fact

contains(asa, alka_seltzer),

the system can infer from the taxonomy that 'asa' is a term of the argument sort 'arg_2_contains', but in this fact is used as a term of the argument sort 'arg_1_contains', which is known to be disjoint from ' arg_2_contains'. The system informs the user about this contradiction and points out that the fact would be compatible if reversed. If the user insists that the fact is well-sorted, this will lead to an appropriate change within the model (see transition from fig. 4 to fig. 5).

Similarly, the system recognizes if in a rule a variable binds to disjoint argument sorts. In this case, no term which can be substituted in a well-sorted manner for the variable is known to the system. If a variable is bound to more than one argument of a rule, only the terms which are members of all those argument sorts can be substituted in a well-sorted manner.

For instance, in the rule

$$\text{contains}(X,Y) \wedge \text{affect_pos}(Y,Z) \Rightarrow \text{affect_pos}(X,Z)$$

only terms which are members of the 2nd argument's sort of 'contains' as well as the 1st argument's sort of 'affect_pos' can be substituted for Y, i.e. the terms known to be members of 'act_agent' as well as 'substance'. In our example, these are all members of 'act_agent' because 'act_agent' is a subsort of 'substance'.

In general, a variable at different argument places can only be substituted by terms of the intersection of these argument place's sorts. The intersection of several argument sorts in our lattice is the infimum of equivalence classes of these argument sorts. The extension of these classes' infimum of the taxonomy is the maximal substitution domain for a variable to be bound to these argument places.

The restriction of variable binding to various arguments of a rule is also used to reduce the set of rule hypotheses generated by BLIP's learning approach. The taxonomy is used to eliminate those hypotheses which cannot be verified according to the fact-base. A rule cannot be instantiated by facts if one variable of the rule is bound to disjoint argument sorts because the taxonomy is made by sorting all terms of all facts of the fact-base. There is no combination of facts with the same term at the argument places corresponding to the variable in the rule. The rule is then a hypothesis which cannot be verified by the known facts. Therefore, the hypothesis should not be generated and tested. If we assume a threshhold of n instances for a hypothesis to be verified, we need not generate and test any hypotheses with fewer than n terms in the intersection of one variable.

Taxonomic rules such as, e.g.

$p(X) \Rightarrow q(X)$,

$p(X) \wedge q(X) \Rightarrow r(X)$, or

$p(X,Y) \Rightarrow q(X)$,

can be verified directly by the sort taxonomy. The only thing necessary to do is to map terms of positive and negative one-place facts to seperate 'sorts'. One-place predicates are then treated differently from those with more argument places. This can be justified by their different meaning.Predicates with more than one argument place express a relation between the terms. One-place predicates in a many-sorted logic divide the set of all terms into three sets:

- the set of terms to which the predicate cannot be applied,
- the set of terms to which the predicate can be applied and is true, and
- the set of terms to which the predicate can be applied and is false.

For instance, the predicate 'red' is true for all red objects, false for all not-red objects, and not applicable to not-objects. By computing in addition to the argument sorts also the sets of terms of positive and negative facts, our procedure can ascertain the validity of the corresponding taxonomic inferences.

The acquisition of a sort taxonomy in BLIP described here is implemented on a Symbolics LISP-Machine in Prolog.

5 Conclusion

In this paper, an implemented approach to the incremental, automatic acquisition of a taxonomy based on the sorting of terms has been presented. The approach is based on the assumption that the user forms sensible expressions - regardless of whether the inputted facts themselves are true or false. The system therefore indicates to the user when terms are used incorrectly with respect to the knowledge the system gained so far, but does not, however, prevent the user from revolving the model acquired so far. By contrast, the system automatically revises the taxonomy if necessary because of new facts.

Because of the set theoretical basis of the procedure, it is independent of the sequence of inputs.

The procedure takes as elementary units positions of arguments. It forms equivalence classes and computes the inclusion and intersection of the classes. The three levels (terms, argument positions of predicates, and classes) provide for an efficient implementation.

References

[BoB 87] J.H. Boose, J.M. Bradshaw: "Expertise Transfer and Complex Problems: Using AQUINAS as a knowledge acquisition workbench for knowledge-based systems", in: IJMMS, Vol. 26, Nr.1, 1987, pp. 3-28.

[BrS 85] R.J. Brachman, J.G. Schmolze: "An Overview of the KL-ONE Knowledge Representation System", Cognitive Science, Vol.9, 1985.

[BPL 85] R.J. Brachman, V. Pigman Gilbert, H.J. Levesque: "An Essential Hybrid Reasoning System", Proceedings IJCAI-85, pp.532-539.

[Hen 77] G.G. Hendrix: "Human Engineering for Applied Natural Language Processing", Proceedings IJCAI-77, pp.183-191.

[KoG 86] Y. Kodratoff, J.-G. Ganascia: "Improving the Generalization Step in Learning", in: [MCM 86], pp. 215-244.

[KRO 88] K. Morik (ed.): "Knowledge Representation and Organization in Machine Learning", Springer, to appear 1988.

[MCM 83] R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): "Machine Learning - An Artificial Intelligence Approach, Vol. I", Tioga Publ. : Los Altos, 1983.

[MCM 86] R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): "Machine Learning - An Artificial Intelligence Approach, Vol. II", Morgan Kaufmann Publ. : Los Altos, 1986.

[MiS 83] R.S. Michalski, R.E. Stepp: "Learning from Observation: Conceptual Clustering", in: [MCM 83], pp.331-363.

[MiS 86] R.S. Michalski, R.E. Stepp: "Conceptual Clustering: Inventing Goal-Classifications of structured Objects", in: [MCM 86], pp.471-498.

[Mon 87] I. Monarch: "The Role of Ontology in Concept Acquisition for Knowledge-Based Systems", in: Proceedings of the 1. European Workshop on Knowledge-Acquisition for Knowledge-Based Systems, 1987.

[Mor 87] K.Morik: "Acquiring Domain Models", International Journal of Man-Machine Studies, 26/1, 1987.

[Mor 88] K. Morik: "Sloppy Modeling", in: [KRO 88].

[MUB 83] T. M. Mitchell, P. E. Utgoff, R. Banerji: "Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics", in: [MCM 83], pp. 163-190..

[Obe 62] Oberschelp: "Untersuchungen zur mehrsortigen Quantorenlogik", in: Mathematische Analen 145, 1962, pp. 297-33.

[Thi 88] S. Thieme: "The Acquisition of Model-Knowledge for a Model-Driven Machine Learning Approach", in [KRO 88].

[Wro 88] S. Wrobel : "Demand-Driven Concept Formation", in: [KRO 88].

Knowledge Acquisition from Texts

by Jens-Uwe Möller, University of Hamburg, FB Informatik - WSV
Bodenstedtstr.16, D-2000 Hamburg 50, West Germany

Abstract

The analysis of written texts is the first step of any knowledge acquisition process. Book knowledge must be analyzed and structured in a way that can serve as a basis for further knowledge acquisition (e. g. as pre-knowledge for stuctured interviews). Thus automating text analysis is an important step toward automatic knowledge acquisition. Interpretation models, reflecting the conceptual structure of an expert system, form a classification scheme for elements of domain-specific knowledge. Based on this classification scheme a method is introduced, to select relevant knowledge elements. At first the coding system PROPOS supports the transformation of a text into a meaning representation. The linguistic-based method EPISTOS selects relevant knowledge elements from this meaning representation, taking their meaning context into account. EPISTOS incorporates as a theoretical concept pragmatic fields, which enable the description of criteria to select classes of knowledge elements provided by interpretation models. It will be described how to establish pragmatic fields and how to adapt them to special demands, like domains, authors or text.

1. Introduction

The analysis of text is an important task in the process of knowledge acquisition for expert systems. This analysis is necessary for the first step, the analysis of domain texts (e. g. from books), as well as for the analysis of expert interview protocols in the further process of knowledge acquisition. The proceeding described here is restricted to descriptional texts containing matter-of-facts (e. g. texts from books). Thus typical problems appearing in dialogues (like high number of elliptic solutions or indirect speech acts) are excluded. Instead the analysis can concentrate on the meaning expressed in a text.

Assuming, that the task of text analysis is to identify relevant elements of knowledge in the text, two problems arise: First it is necessary to specify the nature of knowledge elements, which are relevant for expert system design, and that we will search for. Second, mechanisms must be invented, to select the specified knowledge elements from texts.

Breuker & Wielinga's interpretation models [84] are domain-independent models describing the structure of problem solving tasks. They define classes of relevant knowledge elements and they provide a structure over these classes to arrange identified knowledge elements. In connexion with some further representation formalisms [Clancey 85, Bennett 83] they cover the first problem.

This article describes a systematic connexion between natural language text and classes of knowledge elements. This connexion manages to identify particular knowledge elements in the text, that are attachable to the classes of interpretation models. A two-step solution for the second problem is pointed out subsequently.

In a first step the whole text is transformed into a meaning representation, realized by propositions. The coding of natural language texts into this propositional representation proceeds systematically and semi-automatically.

Based on this meaning representation a method has been developed, to select particular linguistic concepts (words, statements) from the text, that represent knowledge elements and that are relevant for the modelling of domain knowledge. These concepts are attached to the structures of a corresponding interpretation model. The method attains these results by specifying meaning contexts for classes of knowledge elements. These meaning contexts are used

as a heuristic selection criterion. To perform the described task, the method unifies knowledge about basic problem solving tasks, as stated by the interpretation models, knowledge about meaning, that is representated in propositions, and linguistic knowledge.

2. PROPOS - Coding a text

2.1 Function

The intention of using propositions to code a text is to represent linguistic concepts in their meaning contexts. Meaning relations between different concepts are covered by the hierarchical structure of the propositional representation. By this way any analysis method will need no longer to deal with surface features of natural language texts, but with concepts, meaning and structurally defined contexts. Anderson [85,p.114] proclaimed, that "a proposition is the smallest unit of knowledge". This is one reason to use a propositional representation as a basis for knowledge acquisition.

Patel & Groen [86] already realized the advantages of a propositional representation, and they apply *propositional analysis* to knowledge acquisition. Frederiksen [86] used propositions as a basis for discourse analysis, and Fum [85] profitted by a propositional network for text summarization.

But there is one disadvantage of propositions, already critisized by Patel & Groen: The coding process is time-consuming and thus expensive. Thus measures must be taken to minimize this disadvantage. The representation formalism that shall be introduced here, uses meaning categories that are oriented through relative clauses, which are frequent within German grammar. Thus the transformation process from natural language text to propositions can be supported by grammatical information. PROPOS uses information from morphology, syntax, semantics and the meaning of some linguistic constructions. An essential point in the presented coding process is the complete treatment of function words (closed word classes) relative to their meaning. The reasons are convincingly explained by Morrow [86]: Closed classes words often determine the meaning context of an open class word. Closed classes words and their meaning form linguistic knowledge, that is included in the coding system PROPOS. In addition to this information, PROPOS uses morphological and syntactic information to support the coding process.

Meaning interpretation during the coding process is largely restricted by this linguistic knowledge, and a further step to automated coding is done. In PROPOS the coding is semi-automated, what means, that some meaning interpretation (selecting of meaning categories) still must be supported by user interaction. The developed coding algorithm is well-fitted for automation on the premises, that there are some grammatical and morpho-syntactic information. The prototyp of PROPOS integrates a language knowledge acquisition component as a pre-process of the coding algorithm. This component serves for building up a domain-specific lexicon. The prototyp of PROPOS shows the well-suitness for implementation and the efficiency of the coding process.

2.2 Internal Structure of Propositions

Regarding the structure of the propositions used in PROPOS, we have some arguments related by a predicate. It is similar to the manner of writing in LISP:

```
(predicate argument1 argument2 ...)
```

The predicate is filled by meaning categories and predicative concepts[1] (usually verbs, also participles). Arguments are filled by nominative concepts[2], by references to other concepts, by complements or by adjectives and adverbs. In the simplest case, arguments are words from the text surface, which are subsequently designated as *text objects*. Consequently a proposition is a relation over text objects. A proposition can be regarded as a composed object, whereas text objects are atomic.

Some definitions of the propositional representation should now be exemplary presented. The representation follows in an extended Backus-Naur-Form (BNF), whereby the angular mean clamps *optional* and the curved *any repetition*.

1. example

<istein-prop> ::= ISTEIN <element> <group> is-a-prop

Let us first of all consider the ISTEIN proposition. It corresponds, in its own signification, to the generally known *is-a* relation. ISTEIN forms the predicate. The meaning of the arguments, and their relationship together are fixed in accordance with the definition. ISTEIN propositions con be formed in consequence of the auxiliary verb *sein* (to be), in consequence of the prepositions *von* (of) and *unter* (by), as meaning interpretation of genitive attributes and appositions, due as well as to some conjunctions.

2. example

<verb-prop> ::= <verb>	<agens>	agent (nominative case object)
	[<dativobjekt>]	dative case object
	[{<akkusativobjekt>}]	accusative case object
	[<genitivobjekt>]	genitive case object
	[<präpositionalobjekt>]	prepositional object

Verb propositions take an exceptional position, because a meaning category does not form the predicate in them, but a verb does. The meaning of the individual arguments are not fixed by any meaning category, but are determined by the verb in question.

2.3 Meaning Categories

After this introduction to the structure of propositions, the various meaning categories should now be presented and their meaning quickly explained.

Modal Auxiliary Categories

FÄHIG	abilitiy
MÖGLICH	possibility
MÜSSEN	necessity, recommendation
WOLLEN	desire, intention, will
SOLLEN	order, demand, expression of extraneous will
DÜRFEN	allowance

Auxiliary Categories

WERDEN	process of development
HABEN	membership, possession
EX	existence

1 Predicative concepts [Miller 76] describe a deed, an action or an occurance.

2 Nominative concepts describe physical existing objects or abstractive concepts.

ISTEIN membership to a group
REF identitity

predicatives concepts

AKTION[1] predicatives concept as a noun
ZUSTAND adjectively used predicative concept (present participle)
(VERB)[2]

sentence / statement modification

NEG negation
KON subjunctive mood
IMP imparative mood
FRAGE question
TEMPUS tense

more complex constructions

VERGLEICH comparison of two propositions
ODER connects propositions, that derive from term and sentence disjunctions
UND connects propositions, that derive from term and sentence konjunctions

complements

ORT location, source, origin, context of action (e. g.: at the patient)
ZEIT moment, time interval,
frequency (e. g.: *mostly* young women),
time restrictions (e. g.: no longer)
MOD qualities, attributing by properties (big-small)
graduating (very-hardly)
ANZAHL attributing by scales (set, number, entity)
(not scales of location and time)
indefinite items, specifications (anything, others)
MITTEL instrumental complement

causal categories

GRUND cause, reason, purpose, consequence
BEDINGUNG condition
EINRÄUMUNG counter-reason or counter-circumstance

The grouping of meaning categories serves as an improved synopsis, but nothing is said about representation rules. Auxiliary verbs, for example, can be representated under certain conditions by a modal verb category.

2.4 Propositional Network

Propositions are connected to each other during the process of coding a text. The resulting structure reflects the meaning coherence between those propositions. When coding one sentence, the central statement forms the principal proposition. The whole principal proposition, as well as each of its arguments, may be completed or modified by following propositions, as it is

1 AKTION is a pre-predication, which just serves to differenciate from normal verb proposition. When this pre-predication is later considered as a significat category, then interesting analyzing possibilities arise, when taking care, that the succeding structure elements form the real predicate.

2 In the implementation of PROPOS, all of the verb propositions will be criterioated with this pre-predicate.

shown by example in 2.5. The concatenation of propositions, which do originate from one sentence, results in a tree-structure, of whom the root is the principal proposition. Connexions between different sentences get out of this tree-structure, thus inducing a meaning network.

2.5 Coding of a Sample Sentence

The representation with propostions should firstly be illustrated in the following sentence.

Peter kauft bei Edeka einen Apfel. Peter buys an apple at Harrods.
P1 (kaufen Peter Apfel)
P2 (ORT P1 bei-Edeka)

The verb *kaufen* (to buy), together with *Peter* and *Apfel*, forms the core statement of the sentence. It is representated by the verb proposition with the number P1. After the verb follows the handler, the *agent*, followed by the object, on which the handling is being executed (*patient*).

As well as the core statement the sample sentence contains another locational complement. This is coded in an additional complement proposition, the predicate of which is the meaning category ORT. In P2 a reference to P1 follows the meaning category as the first argument. This means that the following complement refers to P1. The complement itself forms the second argument. Let us take another example.

Peter soll einen Apfel kaufen. Peter shall buy an apple.
P1 (SOLLEN Peter P2)
P2 (kaufen Peter Apfel)

A modal verb proposition is being formed here with the meaning category SOLLEN.

2.6 PROPOS for Knowledge Analysis

Both of these simple examples have mediated an impression of the propositional representation. Let us just refer back to check what the representation with propositions means. In a proposition several arguments are being joined together by a predicate. Due to the different structure of various propositions, and the various meanings of the individual arguments, we obtain a well-structured representation of the meaning of an individual sentence and explicit connexions between the sentences of the text. Individual text objects are integrated in the representation as atomics. The coding of a sentence results frequently as a clear tree-structure, the root of which is the principal proposition of a sentence (core statement), to which the respective complements refer to. Therefore propositions are meaning relations between text objects.

For knowledge analysis the propositional representation of the text offers the following advantages.

Firstly knowledge analysis needs not to deal with complex natural language text anymore, but it can base on a well-structured representation formalism. However, a foundation of the analysis onto the text is guaranteed by the preceeding systematical production of propositions. Due to the use of propositions, as basis of the representation, we obtain formal presentations of the meaning printed out explicitly in a text. The predicate determines the meaning of the relationship between arguments. Knowledge analysis based on this formalism, is consequently an analysis over meaning relationships between text objects.

The interpretation of individual arguments of a proposition are thereby fixed, i. e. the meaning is registered in structures. This is very important for knowledge analysis, because it is possible to distinguish or to classify text objects according to the place of the appearance in a proposition.

However, it is not only the structure inside a proposition, but also the set systematical arrangement of the propositions between each other, which serves for structural representation of meaning. Knowledge analysis can also follow relation structures between propositions, and therefore their representated meaning. The proposition has however, a fundamental unit of meaning representation, another important function: it forms the smallest possible meaningful context of a text object. We obtain first in the scope of text analysis, a means oriented formal fundament for the definition of the notion *context*. On this basis are also the structures between propositions, so that also large contexts can be formally defined overlapping propositions.

Knowledge analysis gains from this formal representation, inasmuch as during coding of a text with the aid of a propositional representation, the individual linguistic concepts are being representated in their meaning context. This gives the possibility to select domain-specific concepts by their meaning context. Furthermore, some meaning categories can serve as a direct access to certain relations, fitting for modelling a domain.

3. EPISTOS

3.1 Epistemological Analysis

"Epistemology is the field of philosophy that deals with the nature and sources of knowledge." [Shapiro et al. 87, S.280]

Epistemology is engaged with as well as other matters, with the knowledge and its description and representation. This is of extraordinary interest for the acquisition of knowledge. Natural language is a typical description instrument, in which meaning is being expressed. This meaning should be analyzed in the presented approach and the enclosed knowledge should be transferred to a representation.

The respective working step is signified as *epistemological analysis*, corresponding to the notion knowledge analysis, used above. The main concern is a depth modelling of the knowledge represented in propositions. It is worth forming a model of the domain, which is suitable for the construction of problem solving expert systems, i. e. the model should reflect the conceptual structure [Bennett 83] of the future expert system.

Breuker & Wielinga's interpretation models describe such conceptual structures depending on respective problem solving tasks, e. g. *planning* of *diagnosis*. Interpretation models are not dependant on any specific domain, but they form generic models, which, filled with domain knowledge, lead to a model of the domain. In the scope of the following approach it is of particular interest, that the individual components of interpretation models define classes of knowledge elements. Therefore, interpretation models constitute a criterion for classification and structuring of knowledge elements.

Up to date the task of selecting relevant knowledge elements from a text and arrangeing them in appropriate classes has been performed by a knowledge engineer. This is exactly the starting point of the developed method. EPISTOS shows a way, in order to select knowledge elements and to classify them in regard to a model of the domain. If we assume, that a text contains the know-

ledge, which we intend to model, and that this text has been coded with propositions, then the principal task for EPISTOS is to relate parts of the propositional representation to particular components of an interpretation model. The basic idea of this method leads to the theoretical concept, that describes the existence of pragmatic fields. The utilization of this concept for epistemological analysis is an important step on the way to a (semi-)automated procedure.

3.2 Pragmatic Fields

The central hypothesis of EPISTOS is, that for classes of knowledge elements, like them being defined by interpretation models, typical meaning contexts can be described. It touches the assumption, that various knowledge elements, which arise in similar meaning contexts, belong to some relationship regarding their content. Once these meaning contexts are known, they can then be used as detection criterions, in order to select knowledge elements from a text.

The introduced meaning representation plays an important role in the realization of this procedure, as *contexts* are formally defined. On one hand a natural language text can be represented with the propositional representation, on the other hand, meaning contexts can be defined against propositional structures. The selection of relevant knowledge elements of a domain is reduced on a pattern matching.

Relations between meanings are produced when pattern matching over propositional structures. The co-relations between the meanings touches on pragmatic aspects of language, i. e. like certain linguistic concepts[1] being used together in order to be interpretated in a certain way by the speaker /listener. Pragmatic is understood in the sense of a meaning theory or a text theory.[2]

Miller [78] describes the existence of lexical and semantic fields[1]. In the continuation of these conceptions, the developed theoretical concept should be designed as a *pragmatic field*, because the linguistic concepts as agents of lexical or semantic fields are classed to typical situations. The linguistic concepts get their meaning in a pragmatic context.

Interpretation models build up assumptions of conceptual structures existing in the world. Pragmatic fields show how these assumptions concerning some qualities of the world, reflect in form and content of language. Thus they are an important heuristical instrument: typical relations in language and world are identified and used purposely for knowledge analysis. In order to keep mistakes out, the process of epistemological analysis is secured by interaction; the final interpretative decision is left to the user. The EPISTOS method serves as an *interpretor aid*, in which it selects and classifies.

---

1 A linguistic concept is composed by one or several text object, i. e. it can be for example a word, but also an expression. It is important for the epistemological analysis, that linguistic concepts represent individual knowledge elements.

2 In meaning theory the meaning of a linguistic concept is determined by its correct use. Semantics describes the meaning of a linguistic concept. Since meaning is defined dependant of a context, the border between semantics and pragmatics dissolves. Pragmatic fields touch the assumption, that there are typical contexts for linguistic concepts. The other way around, it means that certain contexts determine the meaning of particular linguistic concepts.
When considering pragmatics as a text theory, then intersentential structures are examined, which refer particularily to the structure and the coherence of texts. With reservations described in 3.8, pragmatic fields can, in which they define many typical contexts for one and the same linguistic concept, serve for the identification of description sequences in the text.
For differentiation between meaning theory and text theory, then see Grewendorf et al. [87].

3 Miller's fields are based on the assumption, that meaning expressed in linguistic concepts can be represented by meaning primitives. A grouping of all primitives forms a conceptual field. One particular word covers a part of this conceptual field with its lexical field. The common semantic field of some words is formed by the union of their lexical fields.

EPISTOS is all-purpose, because it is not founded on domain-specific knowledge, but only on the knowledge on conceptual structures.

3.3 Pragmatic Fields for Epistemological Analysis

The lexical or semantic fields are given by the class descriptions derived from interpretation models. General knowledge serves by defining contexts for these fields. Such a context can, in succeeding working processes, determine other linguistic concepts belonging to the same semantic field like the respective class description. By this method the domain specific linguistic concepts are ascertained.

Pragmatic fields, used for epistemological analysis, contain information on certain propositional structures, in which a semantic field (the class description) occurs.

Given as an example the class designation K, and the following propositions resulting from a text.

(pred1 A B)
(pred2 C D)
(pred2 E F)

The pragmatic field says about K, that K typically results as the second argument in pred2. Therefore, the pragmatic field round K offers a selection criterion to choose further, K-given concepts. In this example they are D + F.

The abstract example shows, how pragmatic fields use the structuring characteristics of the propositional representation. Their efficiency is improved greatly, when we consider, that all elements af a structural description can be classes over concepts. For the use as selection criterion, the classes (with their elements) must be pre-suppositioned. For example, pred2 can be a class designation for pred2a, pred2b, ..., pred2x. This is one important characteristic of pragmatic fields, when they are applied to epistemological analysis.

3.4 Establishing a Selection Criterion

This should be explained on a real example. A ten page chapter from a handbook for asthma patients [Nolte 87] handles the proceeding of diagnosis. One component of the interpretation model for diagnosis describes the class *symptom* (K=SYMPTOM). The lexical field of this class is the German word *Symptom*. It does not appear itself in the sample text, so that at this point progress is impossible without further information. Looking in a dictionary [Duden 74], we see some definitions, which build a semantical field together with the word *Symptom*: *Anzeichen* (indication), *Vorbote* (early sign), *Warnungszeichen* (warning sign), *Kennzeichen* (sign), *Merkmal* (characteristic). However, none of these words are used in the text either. The text has not offered us so far any clue, to search for symptoms.

General knowledge about symptoms tells us, that symptoms must be perceived. From this we derive a class, which builds the predicate of a proposition, because it covers predicative concepts (verbs): pred2=WAHRNEHMEN (perceive). All perception verbs belong to the class WAHRNEHMEN, like *sehen* (see) or *hören* (hear). The respective selection criterion is: (WAHRNEHMEN * SYMPTOM). The success rate of this selection criterion is quite acceptable. Following up proposition are selected when used on the coded example text. Only the *Granulozyten* are not directly interpretable as a symptom.

bemerken notice
P32: 1 (VERB bemerken$_{W32:11}$ LESER Fuß$_{W32:10}$)
6 (ZUSTAND geschwollen$_{W32:9}$ Fuß$_{W32:10}$)

empfinden perceive
P10: 1 (VERB empfinden$_{W10:6}$ * Atmung$_{W10:7}$)
6 (ZUSTAND quälen$_{W10:10}$ Atmung$_{W10:7}$)
P12: 1 (VERB empfinden$_{W12:2}$ *PERSON Angst$_{W12:4}$)
2 (MOD Angst$_{W12:4}$ schrecklich$_{W12:3}$)
P46: 7 (VERB empfinden$_{W46:34}$ Patient$_{W46:33}$ Atemnot$_{W46:2}$)

hören hear
P50: 1 (VERB hören$_{W50:4}$ Arzt$_{W50:6}$ Atemgeräusch$_{W50:12}$)
P51: 2 (VERB hören$_{W51:2}$ *PERSON Pfeifen$_{W51:4}$)
P51: 3 (VERB hören$_{W51:2}$ *PERSON Giemen$_{W51:16}$)
P51: 4 (VERB hören$_{W51:2}$ *PERSON Brummen$_{W51:31}$)
P52: 3 (VERB hören$_{W52:10}$ * Geräusch$_{W52:2}$)

sehen see
P138: 7 (VERB sehen$_{W138:17}$ *PERSON Granulozyten$_{W138:23}$)

verspüren feel/perceive
P35:15 (VERB verspüren$_{W35:50}$ Patient$_{W35:48}$ Atemnot$_{W35:49}$)

Symptoms resulting from the sample text were:
- geschwollener Fuß (swollen foot)
- quälende Atmung (agonizing breathing)
- schreckliche Angst (terrible fear)
- Atemnot (shortness of breath)
- Atemgeräusch (respiratory sounds)
- Geräusch (sound)
- Pfeifen (whistle)
- Giemen
- Brummen (growling)

Taking this list in comparison to all the symptoms contained in the text, one can see that 1 in 3 are known. The knowledge of the symptoms already found extends the possibility of establishing further selection criteria. The proceeding is a kind of *bootstrapping* when stating or adapting pragmatic fields.

3.5 Linguistic Knowledge

We have already seen in the above example, that class forming is very productive for epistemological analysis. Verbs, which always build a predicate in the propositional representation, play a very special role. They take a hermaphroditic position, because they are a variation of the meaning categories, that is unlimited in number. Not only the relations between following arguments are determined, but it is also possible to classify verbs into semantic fields. Once the meaning of the arguments in a verb proposition is fixed for a certain verb, then we receive the *case frames* from Fillmore [68]. When grouping verbs together in a semantic field, we find ourselves on the way to a *verb classification*, like eg. Ballmer & Brennenstuhl [86, 81][1] have offered.

An verb classification oriented through the verbs meaning is particulary interesting for epistemological analysis. Not only the possibility is offered, to attach comparable *processes* to each other, but also the arguments determined in meaning can be classified. Processes, which primary have a different

1 Ballmer & Brennenstuhl normally define the interpretation of verbs in regard to their content as processes. When one has to classify verbs in accordance with their meaning, then these verbs must be grouped together, which mean or signify the same or similar. As the verb now typically signifies the process, we must group such verb together, which signify processes, which totally or partially correspond. [86, p.31] The authors developed a similar classification for English verbs [81].

meaning, but whose contents complement each other, can be bound together (live = born .. die). Ballmer & Brennenstuhl's *process models* have proven to be very helpful for the fixing of classes, utilised in pragmatic fields.

The verb does however, have a further important function: quite often there are verbs, which can be mapped directly onto particular domain-specific relations. The importance of verbs is seen clearly in the example in 3.7.

3.6 Dynamic Pragmatic Fields

The question of how we come to pragmatic fields is of centaral concern in the scope of knowledge acquisition. We have already classified knowledge elements by the aid of pragmatic fields, and now we are turning to the derivation of pragmatic fields, using the knowledge on elements of one class as pre-knowledge for hypothesis forming.

Given, that the class designation is K, and D and F are domain-specific instances, which belong to the semantic field specified by K. Now when comparing the structures of the propositions determined by D and F, we can see that they can be presented by SC = (pred2 * K)[1]. We now have the first selection criterion for a pragmatic field for K. Exactly this described procedure is of immense importance in the beginning of work with pragmatic fields. Sometimes the class name or a synonym, appearing in any proposition, like K in (pred2 G K), helps further, in order to form suitable hypotheses for selection criteria. Finally, the hypotheses must be empirically verified.

The forming of hypotheses becomes difficult - following the abstract example - as it must not always be a pred2-proposition, in which K occurs, and it is also not necessary, that K forms the second argument in all pred2-propositions.

The continuous extension and specification of pragmatic fields is necessary in order to cope with such problems. The description of meaning structures overlapping several propositions and the class forming are ways to specify selection criteria. A pragmatic field can be extended by any amount of selection criteria. Finally pragmatic fields can always be adapted in a way (by alternated hypothesis forming and use), that they clearly select all relevant knowledge elements existing in a particular text. However, then they loose their epistemological general validity and become text-specific. In order to use pragmatic fields in a tool for knowledge acquisition, they must substantially remain general.

3.7 Some Examples for Pragmatic Fields

It can be shown, that the developed method EPISTOS is suitable to model a domain *world*. That is, why some considerable features of a *world* are presented subsequently, which were derived from the sample text, introduced in 3.5. In this paper it is not intended to form a complete model of that domain, the text does not contain all the necessary knowledge anyway, but it will be demonstrated, that EPISTOS offers possibilities to achieve this model. In the task of *diagnosis* the *objects* SYMPTOM and TEST and *causal knowledge sources* are studied.

a) Symptoms

An interesting characteristic of the class SYMPTOM is that the first four and therefore the essential selection criteria (see below) can be assigned to the *information model* in Ballmer & Brennenstuhl's verb classification. The prag-

1 SC = selection criterion, * = any. The designations refer to the example in 3.3.

matic field for SYMPTOM contains, after analyzing the sample text, the following selection criteria. The selected verbs are added to every criterion.

1. (WAHRNEHMEN * SYMPTOM)		perceive
hören	5/5[1]	hear
empfinden	3/3	perceive
bemerken	1/1	notice
sehen	0/1	see
fühlen	2/2	feel
verspüren	1/1	feel/perceive
2. (BEOBACHTEN * SYMPTOM)		observe
beobachten	1/1	observe
abhören	1/1	listen in
hineinhorchen	1/1	listen into
3. (ERLEBEN * SYMPTOM)		experience
erleben	1/1	experience
4. (ERKENNEN * SYMPTOM)		recognize
feststellen	6/6	identify
5. (AUFTRETEN SYMPTOM)		occur
auftreten	7/7	occur
6. (VERURSACHEN * SYMPTOM)		cause
auslösen	1/2	produce(trigger)

b) Test

Particular interest should arouse by the fact, that the selection criteria 4-10 form a proceeding model, similar to the processes, on which Ballmer & Brennenstuhl [86] based their models. A further investigation should check the question, whether this process model of the class TEST sets description boundaries of a test in a text.(see also 3.8)

The pragmatic field for TEST covers the following selection criteria. What is noticed is that in the domain of medicine values of a test are often identified with the test itself (especially 8 and 9). Thus in the list TEST covers also *values*.

1. (NENNEN * TEST)		name
sprechen von	2/2	talk of
heißen	1/1	call
2. (BEZEICHNEN * TEST)		designate
verstehen unter	1/1	understand by
bezeichnen als	3/4	designate
3. (TESTEN * *)		test
überprüfen	1/1	proof
nachweisen	1/1	point out
durchtesten	1/1	test
untersuchen	2/4	examinate

1 The first number shows the number of propositions in the text, which really lead to the knowledge elements of the required class (SYMPTOM). The second number shows the total number of propositions with this verb, which are selected by the respective criterion.

4. (EXISTIEREN * TEST)		exist
es gibt	2/11	there is
5. (VORSEHEN * TEST)		provide for
verlangen	1/1	demand
vorsehen	1/1	provide for
vornehmen	1/1	intend
verwenden	1/1	use for
sich begnügen mit	1/1	be contented with
verzichten	1/1	renounce
6. (GENÜGEN * TEST)		be enough
genügen	1/2	
7. (DURCHFÜHREN * TEST)		carry out
durchführen	1/1	carry out
führen	1/1	lead
fortsetzen	1/1	going on
8. (ERFASSEN * TEST)		record
gewinnen	1/1	gain
erfassen	1/1	record
messen	4/4	measure
bestimmen	3/3	determine
registrieren	1/1	register
9. (BEDEUTEN TEST *)		mean
besagen	2/2	imply
bestätigen	1/1	confirm
beweisen	1/1	show
bedeuten	1/1	mean
aussagen	2/2	demonstrate
10. (AUSFALLEN * TEST)		turn out
ausfallen	1/1	turn out

c) »Causal Knowledge Sources«

The »causal knowledge sources« as described in the interpretation model for *diagnosis* correspond to the etiological hierarchy, introduced by Clancey [85].

There follows some causal relationships between illness and symptom, resulting from an analysis of the class KRANKHEIT (illness), which because of space reasons, we are not able to fully detail here.

1. (AUSWIRKUNG-SEIN SYMPTOM KRANKHEIT)		resulting from
beruhen auf	1/2	based on
ausdrücken	1/1	express
2. (AUSWIRKUNG-HABEN KRANKHEIT SYMPTOM)		resulting in
bedeuten für	1/2	mean
3. (ZU-TUN-HABEN-MIT SYMPTOM KRANKHEIT)		be connected with
zu tun haben mit	3/3	

In the propositional representation, apart from this special verbs, causal relationship is mainly represented by the meaning categories GRUND (reason) or BEDINGUNG (condition). Considering the meaning of the arguments, we should be able to find in the coding of the sample text structures like

(BEDINGUNG SYMPTOM KRANKHEIT) (CONDITION SYMPTOM ILLNESS)

or

(GRUND SYMPTOM KRANKHEIT). (REASON SYMPTOM ILLNESS)

These structures do not exist in the example text, however the tree-like structure of the propositional representation of a sentence can be used. The demand, that SYMPTOM or KRANKHEIT occurs as an argument will be applied to the whole tree section allocated to this argument. The selection succeeds, when an element of the desired class occurs in any proposition of the relevant tree section. This modification of arguments will be marked by # in the selection criterion. We receive as further criteria for »causal knowledge sources«

4. (BEDINGUNG #SYMPTOM #KRANKHEIT)
5. (GRUND #SYMPTOM #KRANKHEIT)

Assuming that we know all symptoms and illnesses from previous analyzing steps, then we receive for this selection eight correct results from eight possibilities. A further causal relationship, contained in the text, is the derivation of symptoms from the processes, they are based on. The corresponding selection criteria are:

6. (HERVORRUFEN * #SYMPTOM) induce
   auslösen 2/2 evoke
   erzeugen 1/1 produce
   produzieren 1/1 produce

7. (ZUSAMMENHÄNGEN * #SYMPTOM) the order of arguments is arbitrary
   (ZUSAMMENHÄNGEN #SYMPTOM *)
   zusammenhängen-mit 2/2 connected with

The previously introduced pragmatic field forms an effective tool to investigate »causal knowledge sources«. Now it is proven that the introduced method not only obtains good results in determining »objects«[1], but is also useful for »knowledge sources« (rules) to be selected from a text.

3.8 Identification of Description Sequences

Up to now, we have covered the individual selection criteria of pragmatic fields, independent of each other. However, pragmatic fields can be more than just a collection of individual selection criteria. The individual criteria, representing meaning contexts, can be structered in a way, that they represent the typical structure of a description of the corresponding linguistic concept. It seems to be possible to identify description sequences for particular linguistic concepts. This should be exemplary demontrated with the tests, which are detailly describes in the sample text.

In order to be able to cover the structure of the description of a test, those selection criteria, which show a similar meaning, are grouped together.

group NAME: 1. (NENNEN * TEST) name
2. (BEZEICHNEN * TEST) designate

group EXISTENCE: 4. (EXISTIEREN * TEST) exist

group PRECONDITION: 5. (VORSEHEN * TEST) provide for
6. (GENÜGEN * TEST) be enough

1 »Objects« and »knowledge sources« are two of the epistemological primitives introduced by Breuker & Wielinga . The other primitives are are »models«, »structures« and »strategies«.

group	CARRYING-OUT:	3. (TESTEN * *)	test
		7. (DURCHFÜHREN * TEST)	carry out
		8. (ERFASSEN * TEST)	record
group	RESULT:	9. (BEDEUTEN TEST *)	mean
		10. (AUSFALLEN * TEST)	turn out

In a text in which tests are described, at least the last four groups must show roughly the named sequence. The naming (NAME) is free in their position. If two or more groups occur in the same sentences, then their sequence is arbitrary. We study the sequence of *Lungenfunktionsprüfung* (lung function examination) with the subordinated tests of determining *Vitalkapazität* (vigoruos capacity) and *Sekundenkapazität* (second capacity).

Lungenfunktionsprüfung (lung function examination)

PRECONDITION P61: 8 (VERB verlangen$_{W61:37}$ Erfassung$_{W61:32}$ Durchfuhrung$_{W61:39}$)
CARRYING-OUT P63: 2 (ZUSTAND gewinnen$_{W63:3}$ Lungenfunktionswert$_{W63:4}$)

Vitalkapazität (vigoruos capacity)

NAME P63: 4 (VERB heißen$_{W63:5}$ Lungenfunktionswert$_{W63:4}$ Vitalkapazitat$_{W63:6}$)
CARRYING-OUT P65: 1 (AKTION Bestimmung$_{W65:2}$ * Vitalkapazitat$_{W65:4}$)

Sekundenkapazität (second capacity)

PRECONDITION P68: 2 (VERB vornehmen$_{W68:7}$ Arzt$_{W68:2}$ Untersuchung$_{W68:6}$)
CARRYING-OUT P69: 1 (ZUSTAND messen$_{W69:2}$ Wert$_{W69:3}$)
NAME P69: 2 (VERB bezeichnen$_{W69:24}$ Wert$_{W69:3}$ Sekundenkapazitat$_{W69:6}$ -als)
NAME P69: 4 (VERB bezeichnen$_{W69:24}$ Wert$_{W69:3}$ Atemstoß$_{W69:19}$ auch_als)
NAME P69: 5 (VERB bezeichnen$_{W69:24}$ Wert$_{W69:3}$ Tiffeneau_Wert$_{W69:23}$ auch_als)

CARRYING-OUT P72: 2 (AKTION Bestimmung$_{W72:2}$ * Wert$_{W72:5}$)

other subordinated tests

CARRYING-OUT P73:10 (AKTION Bestimmung$_{W73:18}$ * Blutdruck$_{W73:20}$)
CARRYING-OUT P73:11 (AKTION Untersuchung$_{W73:32}$ * Urin$_{W73:34}$)
CARRYING-OUT P73:12 (AKTION Registrierung$_{W73:37}$ * Elektrokardiogramm$_{W73:39}$)

RESULT P74:10 (AKTION Aussage$_{W74:34}$ Vitalkapazitat$_{W74:36}$)
RESULT P74:11 (AKTION Aussage$_{W74:34}$ Sekundenkapazitat$_{W74:39}$)

The description of this TEST shows, that the sequence of the groups is roughly kept. However, the examinations carried out are not yet sufficietly consolidated, in order to derive heuristics from them. The specification and identification of description sequences represents an idea, to extend the concept of pragmatic fields and therefore increase their contributions to epistemological analysis.

4. Conclusion

4.1 Results

Pragmatic fields, which have been developed for epistemological analysis, achieve a connexion between knowledge elements coded by propositions and the conceptual structure of expert systems given by interpretation models. Interpretation models consist of several components, which can each be marked as classes over knowledge elements. Pragmatic fields define meaning contexts for knowledge elements, which are arranged along these classes. These meaning contexts serve for the identification of knowledge elements. The selection of knowledge elements proceeds by pattern matching with meaning contexts represented in propositions and propositional structures.

It is possible to automate epistemological analysis by team work of the systematical coding of text on the one hand, and a description of required knowledge elements based on the same formalism on the other hand. Pragmatic fields function as a heuristic for the selection of knowledge elements. Epistemological analysis can be performed automatically or controlled by interaction, according to the quality of this heuristic. Continued improvements of the heuristic are easily carried out.

The examinations on the example text have shown, how the described method enables us to analyze a text in regard to the construction of an expert system. Fundamental pragmatic fields corresponding to some components of the interpretation model for diagnosis have exemplary been established. The internal structure of pragmatic fields shows, that they are not only a collection of meaning contexts serving as selection certeria, but also that the meaning context together form a description structure. This strengthens the conception, that basic parts of pragmatic fields are applicable to all text, independent of a specific domain or any author. They can be adapted as well to special demands, when used dynamicly.

Descriptional texts, as they are be found for example in books, can be analyzed systematically. The contained knowledge gets available to the knowledge acquisition process. In opposition to other approaches to text analysis[1] the introduced procedure is semiautomatic for the first time. Thus the methods is an important contribution to knowledge acquisition.

4.2 Future Works

A linkup with a mature morpho-syntactical parser would be desirable to increase the efficiency of the PROPOS coding procedure. The meaning interpretation could be statistically supported as well, to submit interpretation suggestions to the interacting person. With sufficient evidence, meaning interpretation could be automatically performed.

Pragmatic fields should be developed and tested in the scope of EPISTOS for the most various interpretation models. The construction of an exact corresponding verb classification meeting the demands of epistemological analysis should be of particular interest. With such a useful system, it would be possible to investigate the identification of description sequences.

4.3 Implementation

PROPOS has already been implemented prototypically, which shows that the chosen concept is successful. EPISTOS is still a method, and thus far no implementation of EPISTOS exists.

---

1 Kuipers & Kassierer [83,84], Clancey [85]

References

ANDERSON, J.R.: Cognitive Psychology and its Implications. 2.Ed., Freeman, New York, 1985.

BALLMER,Th.; BRENNENSTUHL,W.: Deutsche Verben - Eine sprachanalytische Untersuchung des deutschen Verbwortschatzes. Narr, 1986.

BALLMER,Th.; BRENNENSTUHL,W.: Speech act classification. A Study in the Lexikal Analysis of English Speech Activity Verbs; Springer, Berlin, New York 1981

BENNETT, J.S.: Roget: A knowledge-based consultant for acquiring the conceptual structure of an expert system. Stanford Heuristic Programming Project, Memo HPP-1983-24.

BREUKER, J.; WIELINGA,B.: Interpretation of Verbal Data for Knowledge Acquisition, ECAI 84: Advances in Artificial Intelligence; O'Shea,T.(Editor); Elsevier Science Publishers B.V.(North-Holland).

BREUKER, J.; WIELINGA,B.: KADS: Structured Knowledge Acquisition for Expert Systems. Les Systemes Experts & Leurs Applications, 5iemes Journees Internationales, Avignon 1985.

BREUKER, J.; WIELINGA,B.: Use of Models in the Interpretation of Verbal Data, in Kidd,A.; Knowledge Elicitation for Expert Systems: a practical handbook. Plenum Press, New York, 1986.

BREUKER, J.; WIELINGA, B.: Models of expertise, ECAI 1986.

BREUKER, J.; HAYWARD, S.A.; WIELINGA, B.J.: Structured analysis of knowledge, International Journal of Man-Machine Studies, 4/1987, p.487 ff.

CLANCEY, W.J.: Acquiring, Representing and Evaluating a Competence Model of Diagnostic Strategy. In: Chi,M.T.H.;Glaser,R.;Farr,M.: Contributions to the Nature of Expertise, 1985.

DUDEN, Fremdwoerterbuch, Bd.5, 3. Aufl., Mannheim 1974.

FILLMORE, Ch.: "The Case for Case".In: Bach,E.;Harms,R.(eds.) Universals In Linguistic Theory, Holt,Rinehart and Winston, p.1-88, New York, 1968.

FREDERIKSEN, C.H.: Cognitive Models and Discourse Analysis, in: Cooper, C.R.; Greenbaum, S.: Written Communication Annual. Vol 1, Studying Writing: Linguistic Approaches, Beverly Hills, CA.: Sage, 1986.

FUM, D.; GUIDA.; TASSO, C.: Evaluating Importance: A Step Towards Text Summarization; IJCAI '85

GREWENDORF, G.; HAMM, F.; STERNEFELD, W.; Sprachliches Wissen - Eine Einfuehrung in moderne Theorien der grammatischen Beschreibung. Suhrkamp, Frankfurt a. M. 1987

KUIPERS,B.; KASSIRER,J.P.: How to Discover a Knowledge Representation for Causal Reasoning by Studying an Expert Physician. IJCAI '83

KUIPERS,B.; KASSIRER,J.P.: Causal Reasoning in Medicine: Analysis of a Protocol. Cognitive Science 8, p.363-385, 1984.

MILLER,G. : Semantic relations among words. In: Halle,M.; Bresnau,J.; Miller,G.:Linguistic Theory and psychological reality, Cambridge, Mass; p.60-117. 1978.

MORROW, D.G.: Grammatical Morphemes and Conceptual Structure in Discourse Processing, Cognitive Science 10, 1986, p.423-455.

NOLTE, D. : Sprechstunde: Asthma. Graefe und Unzer Verlag, 4.Aufl., Muenchen 1987.

PATEL, V.L.; GROEN, G.J.: Knowledge Based Solution Strategies in Medical Reasoning, Cognitive Science 10, 1986, p.91-116.

SHAPIRO, S.C.; ECKROTH, D.; VALLASI, G.A.; Encyclopedia of Artificial Intelligence; Wiley, Canada 1987

Conceptual Models of Interactive Knowledge-Acquisition Tools

Mark A. Musen
Department of Medical Informatics
Erasmus University
3000 DR Rotterdam, The Netherlands*

Workers in artificial intelligence (AI) have developed many interactive programs that assist in the knowledge-acquisition process. Because of the diverse nature of these tools, it is often difficult to understand how each one relates to all the others. This paper describes a taxonomy for knowledge-acquisition aids that is based on the terms and relationships that a given tool uses to establish the semantics of a user's entries. Such semantic assumptions, or *conceptual models,* have important implications for how a knowledge-acquisition tool is used and to what degree it can assist its users in analyzing new applications at the knowledge level. Furthermore, when the conceptual model of a knowledge-acquisition tool can be made explicit, knowledge engineers can use metalevel programs to edit that conceptual model, creating knowledge-acquisition aids that are custom-tailored for particular applications. One such metalevel tool, PROTÉGÉ, has been developed to edit the conceptual models of programs that acquire knowledge for tasks that can be solved via the method of skeletal-plan refinement.

1. Introduction

For many years, the process of building expert systems has been considered an "art" (Feigenbaum, 1977). Yet despite a lack of formal tradition, principles of knowledge engineering have started to emerge recently. An important advance was Newell's (1982) description of the *knowledge level,* including the observation that knowledge is a capacity for behavior that is distinct from whatever data structures are used to represent that behavior at the *symbol level.* Other investigators have built on Newell's work to characterize generic methods of problem solving that are independent of specific inference engines and of specific application areas (Clancey, 1985; Chandrasekaran, 1986). In this context, knowledge engineering can be viewed as a modeling operation in which an abstraction of some *task* in the world (an application problem to be solved) is described in terms of one or more of these problem-solving *methods* (domain-independent strategies for arriving at a solution).

Concomitant with such theoretical work, a large number of system builders have remained concerned with the practical problems of acquiring and encoding expert knowledge. Often motivated by the pragmatic need to develop and maintain large knowledge-based systems, many of these workers have built computer-based tools to assist in the knowledge-acquisition process. Knowledge engineers and, in some cases, domain experts interact with such tools to construct knowledge bases for solving concrete problems. Although important theoretical results have

* The author currently is on leave from the Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, California 94305-5479, USA.

emerged from the development of such knowledge-acquisition tools, it often has been difficult to pinpoint the specific contributions of the diverse tools that have been described in the literature. Many authors have emphasized detailed descriptions of new programs without attempting to explain how each new system relates to all the others. Commonly encountered distinctions such as "general purpose" and "domain dependent" have done little to clarify how particular tools may help system builders in the modeling of expertise.

This paper presents a framework by which interactive knowledge-acquisition tools can be compared. Elaborating on a taxonomy introduced in a previous publication (Musen et al., 1987), this paper describes a classification scheme that centers on a program's *conceptual model*—the terms and relationships that a program uses to establish the semantics of a user's entries. In the case of computer-based knowledge-acquisition tools, the conceptual model determines the manner in which users view the knowledge bases that they are creating or editing. This paper points out the three types of conceptual models that interactive knowledge-acquisition tools may adopt: (1) models based on the *symbol-level* data structures required by the target expert-system shell (as in TEIRESIAS; Davis, 1976), (2) models based on a generic problem-solving *method* (as in ROGET; Bennett, 1985), and (3) models based on a set of application *tasks* to be performed (as in OPAL; Musen et al., 1987). Although some knowledge-acquisition tools may not fit neatly into one of these three categories, the taxonomy helps to clarify the role that individual tools can perform in the process of knowledge-base construction. More important, by elucidating the conceptual models of knowledge-acquisition tools, system builders can become more aware of the assumptions that such tools make, and can develop metalevel editors to define and to optimize the conceptual models of such tools. This paper consequently introduces PROTÉGÉ, a program for creating graphical knowledge-acquisition tools with task-oriented conceptual models, as an example of such a metalevel editor.

2. Conceptual Models for Knowledge Acquisition

A conceptual model allows the user of a computer program to understand the meaning of the messages that the program produces and the nature of the data on which the program operates. Thus, the conceptual model is the semantic framework that is shared between a program's users and a program's developers that permits the users and the program to communicate. For example, it would be impossible to use a spreadsheet program such as VisiCalc without appreciating the simple, but powerful, conceptual model that ascribes meaning to the rows and columns in the grid. Without such a conceptual model, a spreadsheet is merely a collection of numbers and boxes. With such a model, however, a spreadsheet suddenly contains *cells* and *columns* and *rows* that have positional and functional interrelationships. Although the terms and relationships of the model may not be formally represented within the program code of the spreadsheet routines, users must nevertheless understand the model's semantics before they can apply a program such as VisiCalc to the solution of their particular applications.

Similarly, a knowledge-acquisition program must adopt a conceptual model that frames the tool's presentation of what a knowledge base contains. Independent of the computational model of expert behavior that a particular knowledge base *represents* (which has also been referred to as a conceptual model; see Brodie et al., 1984), a knowledge-acquisition tool must incorporate a conceptual model that shapes the manner in which the tool itself portrays and captures the knowledge-base contents. The conceptual model thus forms the basis of a language with which both the tool and the tool's user can describe the contents of

a knowledge base. The most common approach has been to display knowledge in terms of the data structures used to encode knowledge at what Newell (1982) refers to as the *symbol level:* production rules, frames, attached procedures, and so on. Alternatively, some tools attempt to view knowledge bases at the *knowledge level.*

Newell viewed the knowledge level as a means to understand the behavior of all intelligent agents, including that of expert systems. He defined the structure of the knowledge level to include the following three abstractions: (1) the goals of an intelligent agent, (2) the actions of which the agent is capable, and (3) the knowledge that allows the agent to carry out actions to achieve its goals. Increasingly, workers in AI are suggesting that defining the behaviors of an expert system at the knowledge level is an important prerequisite to implementing the system at the symbol level. *Knowledge-level analysis* is the term that refers to this modeling of an intelligent system's behaviors, independent of whatever symbols might ultimately be used to program those behaviors in a computer. Although there is not yet consensus on this point, Clancey (1985) has argued that a useful way to perform knowledge-level analysis is to start with the terms and relationships of a well understood model that defines an applicable problem-solving method,[1] such as that of heuristic classification (Figure 1). By describing how the solution to a domain task can be obtained in terms of the given problem-solving method, system builders can perform a knowledge-level analysis of the required intelligent behavior that is independent of any symbol-level implementation.

Knowledge-acquisition tools with knowledge-level conceptual models come in two varieties. The first group of such tools incorporates conceptual models that reflect generic problem-solving methods (domain-independent solution strategies) that the tools' users must apply to individual expert tasks. The second group of knowledge-level tools, on the other hand, incorporates conceptual models of prototypical domain tasks (problem statements) that users must instantiate to define specific applications. We now consider these symbol-level and knowledge-level conceptual models in detail.

2.1. Symbol-Level Conceptual Models

Most knowledge-acquisition tools—from Davis' landmark program TEIRESIAS (Davis, 1976) to the knowledge-base editors used in conjunction with virtually all commercial expert-system shells—present the contents of a knowledge base in terms of symbol-level entities. TEIRESIAS, which attempted to communicate with its user in a subset of natural language, still required the user to view the MYCIN knowledge base as a collection of individual rules. Thus, although the *medium* of interaction was a subset of English, TEIRESIAS asked its user to think about MYCIN's knowledge in terms of rule clauses and of the parameter values that the rules concluded. Even though the rules in the knowledge base collectively defined a problem-solving strategy (namely, heuristic classification in the infectious-disease domain), TEIRESIAS could not address the knowledge from that perspective; instead, the system was designed to focus on the individual rules in a single chain of reasoning.

---

[1] Chandrasekaren (1986) uses the term *generic task* to describe much the same concept.

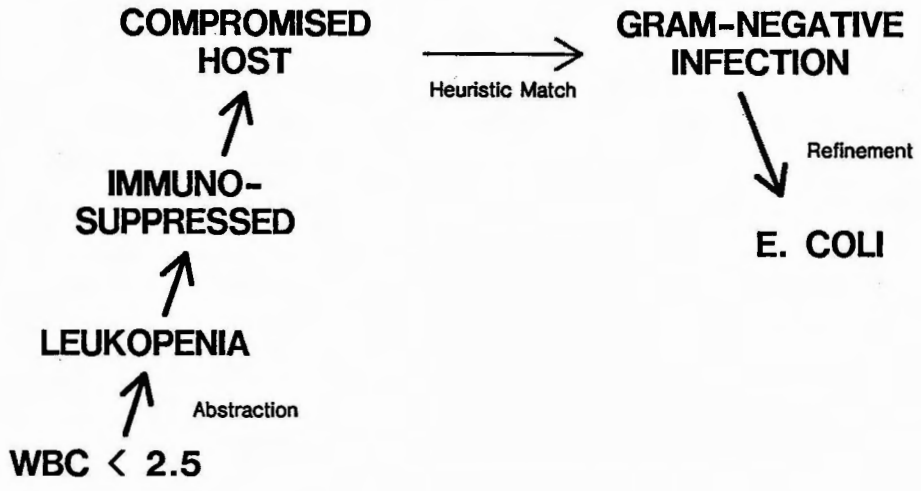


Figure 1: Heuristic classification is a problem-solving method in which a selection is made from a pre-enumerated solution set. Heuristics link feature-abstraction hierarchies to solution-refinement hierarchies. Here, the method is applied to the antibiotic-selection task in MYCIN. (Source: Adapted from Clancey, 1985, p. 295)

TEIRESIAS, like the many current knowledge-acquisition tools that present the contents of a knowledge base in terms of symbol-level entities, inherently ignored the knowledge level. Such symbol-level tools can display only either the data structures in the knowledge base or transformations of those data structures into isomorphic linguistic (Davis, 1976) or graphical (Hannan & Politakis, 1985) notations. Knowledge-level analysis—the process of defining the goals for an intelligent system and of modeling the behaviors required to achieve those goals—is thus left entirely to the user and cannot be supported by the knowledge-acquisition tool. Communication with symbol-level tools consequently requires the user to transform knowledge-level perceptions into terms that match the requirements of the expert-system inference engine at hand. In TEIRESIAS, for example, the natural-language interface could do nothing to insulate the user from needing to understand how MYCIN production rules were invoked via backward chaining. Furthermore, the ability to enter rules in English did not exempt the user from having to understand nuances such as (1) how knowledge engineers crafted special "control rules" to establish the goals for MYCIN's reasoning and (2) how MYCIN's behavior could be influenced in important ways by changing the order of the clauses in the left-hand sides of rules (Clancey, 1983).

All knowledge-acquisition tools with symbol-level conceptual models expect that their users will appreciate both the nature of the representations used to encode the knowledge in the target expert system and the consequences of applying a particular inference mechanism to those symbols. Without such an appreciation, it is impossible to discuss the contents of a knowledge base in symbol-level terms. Consequently, users of knowledge-acquisition tools in this class must understand in large measure how expert systems are implemented.

2.2. Method-Based Conceptual Models

Although symbol-level knowledge-base editors are by far the tools most commonly used for construction of expert systems, a number of workers have begun to experiment with knowledge-acquisition aids that attempt to view

expert-system knowledge bases more abstractly. Rather than discussing an expert system's knowledge in terms of symbol-level representations, users of such tools attempt to communicate knowledge specifications in terms of the *behaviors* that are to be achieved by the target expert system. These knowledge-level behaviors are described for the computer using special-purpose languages that, like all languages, are necessarily composed of symbols.

At first glance, the reliance on symbols for entry of knowledge-level specifications would seem to run contrary to Newell's description of knowledge as an abstraction that can never be written down completely. Yet it is still possible to use symbols to express the results of a knowledge-level analysis when the *meanings* of those symbols can be derived from a prespecified model. Clancey (1985), for example, has shown that knowledge engineers can use the terms and relationships in the *heuristic classification model* (a set of symbols) to record knowledge-level descriptions for a variety of classification tasks. The use of the underlying model to establish the meaning of the symbols contrasts with the approach taken in traditional knowledge engineering, where the semantics of a user's specifications are defined only operationally by the inference engine of the target expert system.

Models of problem-solving such as that of heuristic classification can be used effectively as the conceptual models for interactive knowledge-acquisition tools. In the use of such tools, the human–computer dialog does not center on the rules, frames, or other symbol structures in the knowledge base; instead the interaction concentrates on the methods by which task-specific knowledge is used to arrive at a problem's solution.

Such a method-based conceptual model may be based on some *explicit* set of terms and relationships, such as those of the heuristic classification model. To enter knowledge into knowledge-acquisition tools of this class, the user must apply the terms and relationships of the model to the features of the application task at hand. In the case of heuristic classification, for example, the user would be asked to identify the elements of the *solution set,* the *data-abstraction hierarchies,* the *solution-refinement hierarchies,* and the *heuristics* that associate one hierarchy with another (see Figure 1). A second approach is to shield the user from specific details of the problem-solving method, even though a model of that method drives the behavior of the knowledge-acquisition tool internally; the model of problem solving thus becomes *implicit* from the conceptual perspective of the user. When the problem-solving model is made implicit, the user's interaction with the knowledge-acquisition tool does not need to refer to the specific terms and relationships in that model. This alternative technique requires that the knowledge-acquisition tool often make significant assumptions about the knowledge that a user enters.

2.2.1. Explicit Models of Problem-Solving

The first knowledge-acquisition tool to adopt a method-oriented approach was ROGET (Bennett, 1985), a program designed to assist knowledge engineers and domain experts in developing EMYCIN-based expert systems. Using an explicit model of diagnostic problem solving that can be viewed as a specialized form of heuristic classification, ROGET conducted a dialog with its user, asking what kinds of evidence might be gathered during a consultation and how that evidence might suggest or discriminate among possible problems and their causes.

ROGET's model of diagnosis provided a mechanism by which a knowledge engineer could describe an expert system's intended behavior at the knowledge level. ROGET asked about the *problems* to be diagnosed, about the *causes* of those problems, and about *data* that could be used to suggest, confirm, or rule out those causes and problems. The particular symbols that might be used to *encode* this

information as a knowledge base never entered into the human–computer discussion. A user's dialog with ROGET thus produced a knowledge-level specification of the application, which was then translated into EMYCIN symbols that could form the basis of a working consultation program. The user of ROGET, however, never needed to think about the resulting EMYCIN rule base; instead, the user was asked only to conceptualize the entered knowledge in terms of ROGET's problem-solving model.

There are now a number of computer-based tools that adopt conceptual models for expert-system knowledge that are based on explicit models of problem solving. MORE (Kahn et al., 1985), MOLE (Eshelman et al., 1987), and SALT (Marcus et al., 1985) are all method-oriented knowledge-acquisition aids that are representative of this class. MORE and MOLE both incorporate models of problem-solving methods that are specialized forms of heuristic classification, whereas SALT embodies a "propose and revise" method that *constructs* the solution to a problem. With each such tool, users must learn to describe domain tasks using the terms and relationships of some model. These terms and relationships allow the program to lead the user through a knowledge-level analysis of the application. McDermott (1986) has emphasized that a principal rationale for building tools of this class lies in developing new models of computational problem-solving methods, and in exploring the scope and limitations of those methods when applied to real-world tasks.

2.2.2. Implicit Models of Problem Solving

Not all method-oriented tools require the user to understand the terms and relationships of an explicit model. In particular, a growing number of knowledge-acquisition programs based on personal-construct psychology (Kelly, 1955) solicit knowledge about classification tasks in a manner that largely conceals the problem-solving model from the user.

ETS, for example, is a system that acquires knowledge during the early stages of expert-system development (Boose, 1985). ETS automates the psychiatric interviewing techniques that were originally devised by therapist George Kelly (1955) to learn how people make distinctions in the world. ETS uses a structured dialog with an application specialist to solicit the elements of the *solution set* (the possible classifications that might apply to a given case) and the *features* of a case that might be relevant in arriving at a classification. The program then generates a rating grid—a matrix in which each classification in the solution set is matched with each possible feature. For each classification–feature pair in the matrix, the user enters a number that indicates the degree to which she believes the particular feature is predictive of the indicated classification. ETS analyzes the rating-grid data to determine dependencies among the features (for example, that whenever the feature *leukopenia* is associated with a category of meningitis, the feature *immunocompromised* also is present, but not *vice versa*). ETS generates putative abstraction hierarchies automatically from the implicit dependencies that it observes among the features. The program then converts these abstraction hierarchies into condition–action rules (for example, **IF** the patient has leukopenia, **THEN** the patient is immunocompromised).

The user of ETS, unlike the user of tools such as ROGET, does not need to understand all the details of the program's problem-solving model, because the classification hierarchies in ETS are generated indirectly. The user views the knowledge as a simple table of feature ratings; there is no need to view the knowledge in terms of *hypotheses* and *evidence*. The program—not the user—attempts to fit the rating-grid data into the classification model. The advantage is that ETS can be used by computer-naïve application specialists after very little instruction. However, because the classificatory associations inferred by

the program are frequently incomplete and sometimes spurious, the program is ordinarily useful only during the initial stages of expert-system construction. In a successor program, called AQUINAS, Boose and Bradshaw (1987) have attempted to overcome some of the limitations of ETS by making the classification model more explicit.

Like ETS and AQUINAS, a knowledge-acquisition tool called KITTEN (Shaw & Gaines, 1987) and a recent commercial product called Auto-Intelligence (IntelligenceWare, Inc., Los Angeles, California) also adopt Kelly's interviewing techniques and the model of classification problem solving used in personal-construct psychology.

2.3. Task-Based Conceptual Models

Rather than incorporating models of problem solving methods, some knowledge-acquisition tools instead adopt conceptual models that reflect recurring domain *tasks* to be performed. Defining a particular application then requires that users apply the explicit terms and relationships of a general task model to a specific task instance. For example, in the case of OPAL (Musen et al., 1987)—a graphical tool that allows cancer specialists to enter new treatment plans into the knowledge base of an expert system called ONCOCIN (Shortliffe et al., 1981)—the knowledge-acquisition program contains an explicit model of the components of cancer therapy (Figure 2). Unlike method-oriented programs such as ROGET, OPAL adopts a conceptual model that includes only *domain-specific* concepts such as *chemotherapy, drug, toxic reaction,* and *laboratory test.* Physicians use this task model to enter specific cancer-treatment plans into OPAL. Thus, the terms and relationships in OPAL's abstract model of the cancer-therapy task form the basis for a language that is used by both the program and its physician-users to describe cancer-therapy *instances.* The program's users think in terms of only the particular cancer treatment to be prescribed; the *methods* by which ONCOCIN may determine how to administer particular chemotherapies and drugs are not depicted in the task model, and thus never need to be described for OPAL.

Student (Gale, 1987) and KNACK (Klinker et al., 1987) are two other recent knowledge-acquisition tools with task-based conceptual models. In Student, the conceptual model defines the general procedure for organizing and submitting data for analysis by a statistical package such as SPSS or SAS. Each instantiation of that model by a user of Student yields the knowledge that pertains to a particular data-analysis application. In KNACK, the conceptual model defines the contents of a text report that appraises the design of certain classes of electromechanical devices. Users extend KNACK's model to define how to generate such reports for particular device types.

In all task-based knowledge-acquisition tools, the problem-solving method that the target expert system must apply to a given task is predetermined by the developers of the tool and remains transparent to the user. (In OPAL, for example, that problem-solving method is a form of skeletal-plan refinement; Friedland & Iwasaki, 1985.) All task-based knowledge-acquisition tools must be constructed with a predefined mapping between the task description and the appropriate problem-solving method. The developers of such tools must therefore perform a knowledge-level analysis of the application area *in advance.* Because the knowledge-level analysis has been done, creating new knowledge bases with tools such as OPAL is relatively simple: Users need to supply only the content knowledge that distinguishes one task instance from another. The *methods* by which individual tasks are solved are already built into the system, and thus need not be of concern. Such tools are useful, however, in only those

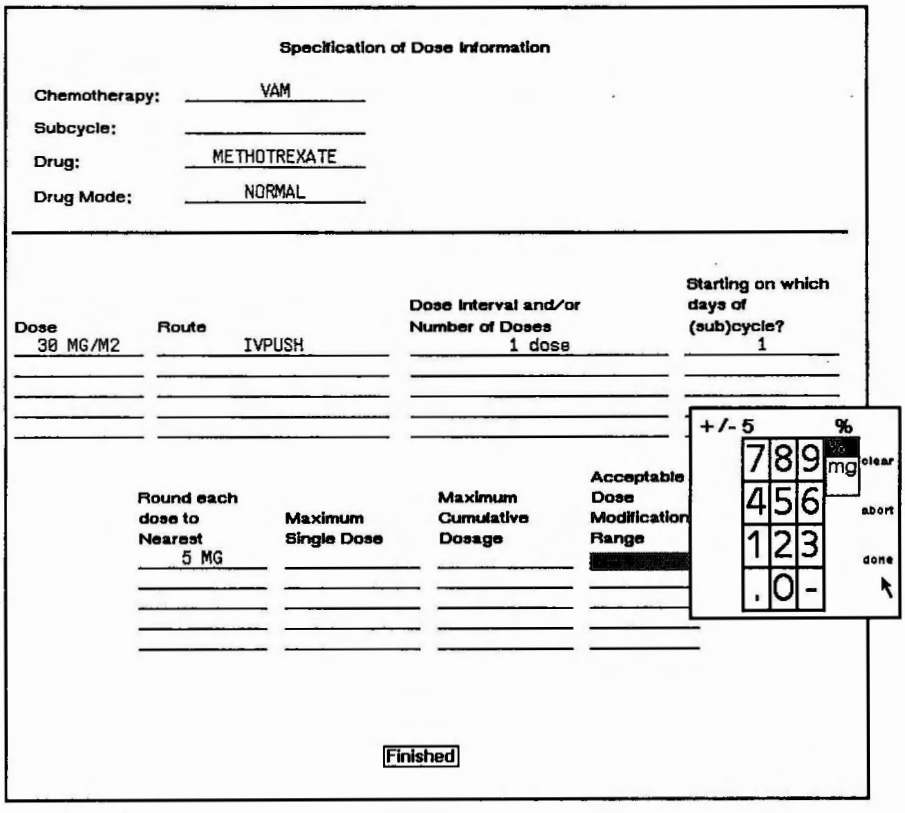


Figure 2: OPAL adopts a conceptual model such that all knowledge is entered in terms of a predefined model of the task of administering cancer chemotherapy. This graphical form in OPAL asks the user to enter the attributes of a drug. In the figure, the user is specifying properties of the drug *methotrexate* when given as part of a drug combination called *VAM.*

application areas where multiple, related knowledge bases are required for analogous domain tasks—that is, in those where each task can be viewed as an instantiation of a common model.

3. Conceptual Models and the Knowledge Level

The large number of interactive knowledge-acquisition aids that have been described in the literature can be classified on the basis of the conceptual knowledge-base model that each program presents to its users. Recognizing whether that conceptual model is based on (1) symbol-level entities, (2) a problem-solving method, or (3) a general task model establishes how knowledge-level analysis is performed for new applications. When the conceptual model reflects symbol-level entities (as does that in TEIRESIAS and most commercial knowledge-base editors), the user of the tool must perform knowledge-level analysis without assistance. When the conceptual model reflects a problem-solving method (as in ROGET), the knowledge-acquisition tool leads the user through a knowledge-level analysis of the application task. When the conceptual model of a knowledge-acquisition tool reflects a general task model (as in OPAL), knowledge-level analysis must be done in advance by the tool's developers.

Some knowledge-acquisition tools may not fit precisely into this taxonomy. It is possible, for example, that both method-oriented programs and task-oriented programs may permit users to enter as symbol-level data structures those concepts that are not expressible in terms of the tools' knowledge-level models.

Similarly, some method-oriented tools may incorporate considerable domain-specific knowledge. The conceptual models of such tools consequently have similarities to those of task-oriented tools; the difference is that additional domain modeling (knowledge-level analysis) must still be done by the users of the method-based programs.

The issue of who is responsible for knowledge-level analysis when developing expert systems is of more than just academic interest. The work required to establish the goals of an intelligent system and to create models of the behaviors by which those goals should be achieved is, in large part, what makes knowledge engineering so difficult. Although symbol-level knowledge-acquisition tools can help to ensure that a user's entries both contain proper syntax and are in some way consistent with prior knowledge-base contents, rule editors and object-oriented languages can do nothing to help the user to formulate knowledge-level models. Even the sophisticated metaknowledge in TEIRESIAS could only point out clauses in particular rules that, on the basis of their syntax, might cause MYCIN to fail. TEIRESIAS could not begin to comment on possible flaws in MYCIN's *model* of infectious diseases. For example, although TEIRESIAS could identify that *organism* was a "context" in the MYCIN knowledge base, the program had no way of knowing what an organism was or what role an organism might play in choosing antibiotic therapy. Knowledge-level analysis was something that TEIRESIAS assumed would be performed by the user "off-line."

On the other hand, knowledge-acquisition tools such as MORE (Kahn et al., 1985) and MOLE (Eshelman et al., 1987), which present the contents of a knowledge base in terms of an explicit, method-oriented conceptual model, actively lead their users through a knowledge-level analysis of the application being encoded. Although the user must still formulate a model for the behavior of the expert system being developed, the knowledge-acquisition tool can prompt for the essential features of that model (for example, the solution set and abstraction hierarchies in the case of heuristic classification). More important, such method-based tools can *critique* a user's entries in terms of the problem-solving models that the tools incorporate, pointing out potential weaknesses in the emerging knowledge base from a knowledge-level perspective. (MORE and MOLE, for example, inform their users when potentially competing hypotheses may not be fully distinguishable on the basis of those features that have already been defined, and point out places in the appropriate abstraction hierarchies where additional features might be relevant to the classification problem.)

Tools that are symbol-oriented or that have method-oriented conceptual models assume that their users will be able to construct models of the applications that are to be encoded. Thus, tools in both these classes require that knowledge engineers exercise creativity in building new knowledge bases. These tools have the versatility to permit development of entirely new applications—which demands that their users (either alone or with the assistance of the computer) attempt to envision the future behaviors of the expert systems that those users are creating.

Knowledge-acquisition tools such as ETS—in which the problem-solving method is not made explicit in the program's conceptual model—still require creativity on the part of the user. The user alone is responsible for identifying features of elements in the application area, labeling those features, and imagining how the features might apply to various classifications in the task domain. The primary difference between ETS and tools such as MORE and MOLE is that untrained users can interact with ETS without having to understand the details of the program's underlying model of classification. Were the problem-solving method to be made more explicit in the ETS conceptual model, users first would have to learn the meaning of the corresponding terms and relationships. Whereas the need to

appreciate the subtleties of a complex model of a problem-solving method often make tools such as MOLE unsuitable for use by novices, the relationships that can be expressed in terms of such explicit models can be considerably richer than are those that can be entered into programs such as ETS. In both cases, however, users must know *a priori* that the particular method of problem solving built into the tool can be applied to the task that they wish their target expert system to address.

Unlike method-based knowledge-acquisition aids, tools with task-based conceptual models generally require not that their users be particularly creative, but rather that they be diligent. Because knowledge-level analysis has already been done by the tool developers, users do not have to elucidate concepts in the application area or to specify the problem-solving behavior of the expert systems that they wish to build. The modeling process that often makes knowledge acquisition so laborious is no longer an issue once a task-based tool such as OPAL or Student has been built. The modeling problems do not go away; they are merely removed from the concerns of the tool user. The bottleneck then becomes to construct the knowledge-acquisition tool itself, rather than to develop all the knowledge bases that the task-based tool can be used to create.

Our experience with OPAL represents a case in point. Before the availability of OPAL, encoding a typical cancer-treatment plan for the ONCOCIN system required between 1 and 3 person-months of work. With OPAL, the same knowledge usually could be entered in approximately one week. (To date, physicians have used the tool to specify over 36 cancer-treatment plans for ONCOCIN.) Despite this significant acceleration in knowledge acquisition, over 3.5 person-years were needed to develop OPAL in the first place. Much of the work in building OPAL involved repeatedly refining the program's conceptual model of cancer therapy, and recoding the system accordingly. The ability to revise that model without the need for reprogramming undoubtedly would have reduced the development time considerably.

4. Editing a Conceptual Model

By viewing knowledge-acquisition tools in terms of their conceptual models, workers in AI can think more critically about the effort required to construct such programs. The relationship between the entries made by a user and the contents of the resulting knowledge base become more apparent. The degree of knowledge-level analysis required of the tool's developers and of the tool's users becomes more clear. Furthermore, when the terms and relationships of a tool's conceptual model can be made explicit, it is possible to build a *metalevel program* that can edit that conceptual model. Such a program then can facilitate the development of a variety of computer-based knowledge-acquisition aids.

PROTÉGÉ (Musen, 1988) is such a metalevel editor. By filling out graphical forms on a workstation screen, knowledge engineers and domain experts can use PROTÉGÉ to define a set of knowledge-level terms and relationships that can model a particular set of application tasks (Figure 3). PROTÉGÉ then creates a graphical knowledge-acquisition program that looks much like OPAL, but that is custom-tailored for the class of application tasks that was defined using PROTÉGÉ (Figure 4). Because PROTÉGÉ itself adopts a conceptual model that reflects the method of skeletal-plan refinement (Friedland & Iwasaki, 1985), only those tasks that can be solved using this particular problem-solving method can be described for the system. Nevertheless, the approach demonstrates an important multiplicative effect: Defining a domain-specific conceptual model

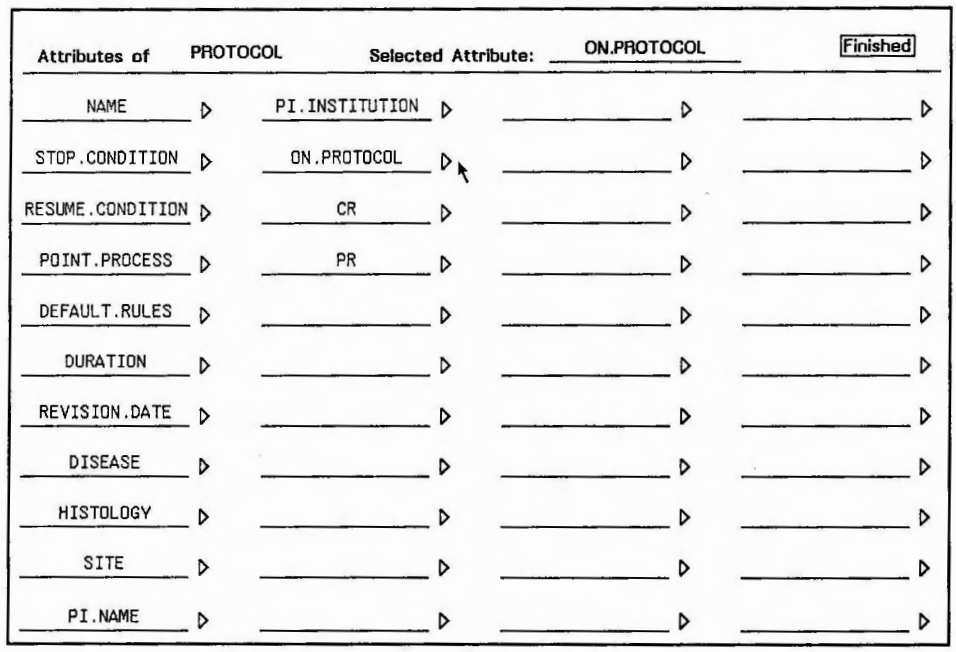


Figure 3: This is one of many graphical forms in PROTÉGÉ that allow knowledge engineers to define conceptual models for new task-oriented knowledge-acquisition tools. This form solicits the attributes of *planning entities* in a class of application tasks. Here the user is entering the attributes of *protocols* in the cancer-chemotherapy domain.

with PROTÉGÉ creates a new knowledge-acquisition tool specialized for a particular class of application tasks; every successful interaction with one of these PROTÉGÉ-generated tools in turn produces a new knowledge base for a particular expert-system shell (called *e-ONCOCIN*). To date, PROTÉGÉ has been used to create two functioning knowledge-acquisition aids: *p-OPAL,* a program that replicates the functionality of OPAL, and *HTN,* a tool for entry of knowledge concerning clinical trials of antihypertensive drugs (Musen, 1988).

PROTÉGÉ, like other method-based knowledge-acquisition programs, leads its users through knowledge-level analyses of relevant application tasks. The tools produced by PROTÉGÉ (task-oriented knowledge-acquisition aids like OPAL) make no knowledge-analytic demands of their users, however. The PROTÉGÉ-generated tools are particularly well suited for use by domain experts, who cannot be expected to have experience in performing knowledge-level analysis on their own. PROTÉGÉ, on the other, hand, specifically assists knowledge engineers with these difficult—and unavoidable—modeling issues.

5. Conceptual Models and the User

Early workers in AI identified the arduous problems of knowledge acquisition simply as problems in communication. The knowledge engineer was viewed as an intermediary who did not understand the application area very well, but who

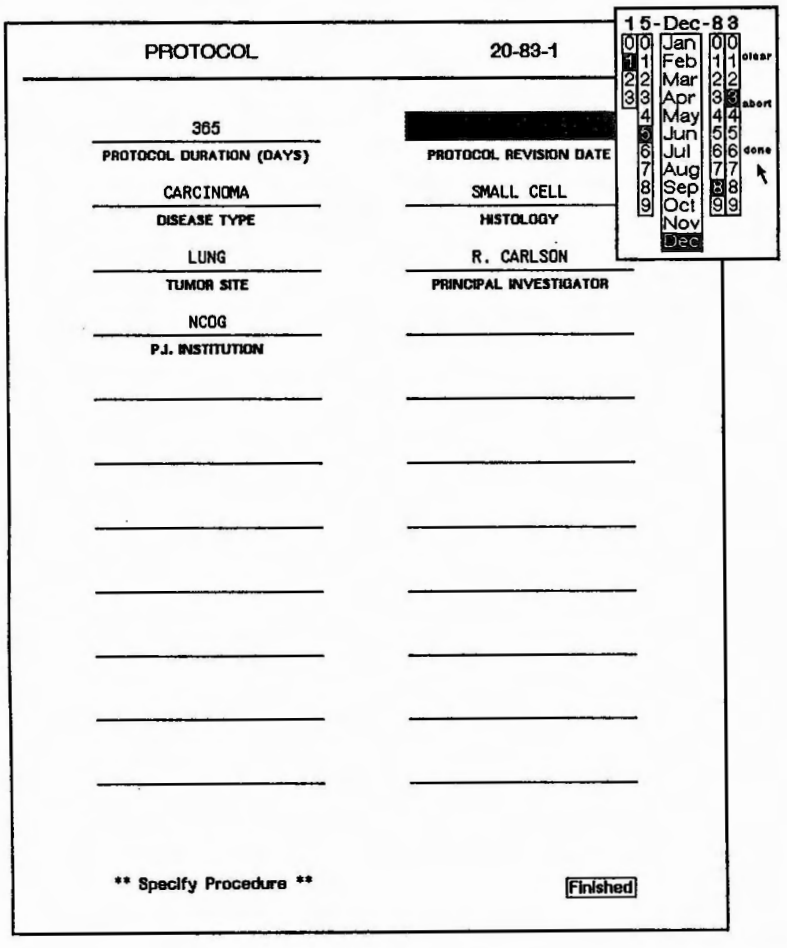


Figure 4: The conceptual models of the knowledge-acquisition tools generated by PROTÉGÉ are tailored to the knowledge engineer's metalevel specifications. This graphical form from a tool called *p-OPAL* solicits information about oncology protocols according to parameters that were defined, in part, using the PROTÉGÉ form in Figure 3. Compare this form with the OPAL form in Figure 2.

served the essential role of channelling an expert's knowledge into the emerging expert system. If knowledge acquisition was slow and laborious, it was felt to be so because the domain expert and the knowledge engineer did not speak the same language. The frequent need for iterative refinement of the knowledge base generally was blamed on communication failures between the two parties (Buchanan et al., 1983).

In the 1970s, research on knowledge acquisition became synonymous with attempts to eliminate the role of the knowledge engineer as "middleman." From the time of TEIRESIAS, builders of knowledge-acquisition tools measured their success largely on the basis of whether their programs could be used independently by application specialists. Indeed, a number of workers have developed symbol-level tools (Tuhrim et al., 1988), method-oriented tools (Marcus et al., 1985; Boose, 1985), and task-based tools (Musen et al., 1987; Klinker et al., 1987) that domain experts have used successfully without assistance. The gold standard by which knowledge-acquisition tools are often judged is simply whether one or more application specialists have actually used a tool to build at least one knowledge base without relying on a knowledge engineer for help.

But knowledge acquisition is more than just the "transfer of expertise." Knowledge acquisition is a creative process in which system builders construct qualitative models of human behavior (Winograd & Flores, 1986; Regoczei &

Plantinga, 1987). Knowledge acquisition necessitates knowledge-level analysis, and knowledge-level analysis for practical applications invariably is difficult. An important and often understated goal of knowledge-acquisition research must be the development of techniques to aid knowledge engineers in their role of helping domain experts to create models of the experts' problem-solving behavior.

Knowledge-acquisition tools that have conceptual models based on the explicit terms and relationships of models of problem solving (programs such as ROGET, MOLE, and PROTÉGÉ) are tools for assisting with knowledge-level analysis. As such, they are best viewed as tools for knowledge engineers. Although it may be feasible and desirable for domain experts to learn to use programs in this class autonomously (see Marcus, 1985), these method-oriented tools are ideally suited to assist knowledge engineers in the creation of knowledge-level models. That these tools may not always be practical for domain experts working alone does not detract from the important benefits that method-oriented tools offer knowledge engineers. The ability to describe an expert system's behavior in terms of a predefined problem-solving method clarifies the knowledge that must be incorporated within the system and helps to elucidate those domain concepts about which additional knowledge is required.

Although there are few data points for making an assessment, the tools that appear to be most successful in allowing application experts with no programming experience to create useful, complete knowledge bases are those with task-oriented conceptual models. Programs such as KNACK, OPAL, and the tools produced by PROTÉGÉ do not require their users to generate knowledge-level models; the users are asked to enter only the content knowledge that defines individual task instances. Although the knowledge that experts can enter into task-oriented knowledge-acquisition programs is thus highly restricted, such task-based tools may be readily constructed for new domains by knowledge engineers using metalevel programs such as PROTÉGÉ. A productive combination of approaches involves the use of method-oriented programs such as PROTÉGÉ to assist with knowledge-level analysis, followed by the entry of content knowledge by domain specialists via special-purpose task-oriented tools (Musen, 1988).

Despite the abstractive power offered by task-based and method-based tools, there may always be a need for traditional symbol-oriented knowledge-base editors in practice. Our knowledge-level conceptual models are, unfortunately, rarely so complete that we can dispense with the need to address certain issues at the symbol level. In OPAL, for example, the program's conceptual model of cancer therapy is occasionally inadequate to allow for entry of atypical treatment specifications. In such circumstances, knowledge engineers often find it more pragmatic to make symbol-level adjustments to the ONCOCIN knowledge base than to make the requisite enhancements to OPAL's task model that would allow physicians to enter that knowledge directly.

What is needed are techniques that will allow knowledge engineers to refine the conceptual models of interactive knowledge-acquisition tools more easily. System builders must be able to experiment with the conceptual models of their development tools to explore the models' limitations and, consequently, the limitations of the knowledge bases that those models can help to construct. Just as there are important advantages to making the problem-solving models in expert systems more explicit (McDermott, 1986), there are benefits to making the conceptual models of knowledge-acquisition tools more adaptable and more distinct. Developers then can design metalevel programs such as PROTÉGÉ to edit the terms and relationships of those conceptual models, tailoring the corresponding tools for particular knowledge-engineering applications.

Acknowledgments

This work was funded by grants LM-07033 and LM-04420 from the National Library of Medicine, and by grant NF-65/62-521 from the Netherlands Organization for Scientific Research (NWO). Computing facilities were provided by the SUMEX-AIM resource under NIH grant RR-00785. Ted Shortliffe, Bruce Buchanan, Larry Fagan, Jim Bennett, Johan van der Lei, and Jan Kors provided many valuable discussions.

References

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

Boose, J.H. (1985) A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man-Machine Studies,* **23**:495–525.

Boose, J.H. and Bradshaw, J.M. (1987) Expertise transfer and complex problems using AQUINAS as a knowledge acquisition workbench for knowledge-based systems. *International Journal of Man–Machine Studies,* **26**:3–28.

Brodie, M.L., Mylopoulos, J., and Schmidt, J.W. (eds.) (1984) *On Conceptual Modeling.* New York: Springer-Verlag.

Buchanan, B.G., Barstow, D., Bechtal, R., Bennett, J., Clancey, W., Kulikowski, C., Mitchell, T., and Waterman, D.A. (1983) Constructing an expert system, in Hayes-Roth, F., Waterman, D.A., and Lenat, D.B. (eds.) Building Expert Systems, Reading, Massachusetts: Addison-Wesley, pp. 127–167.

Chandrasekaran, B. (1986) Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. *IEEE Expert,* **1**:23–30.

Clancey, W.J. (1983) The epistemology of a rule-based system: A framework for explanation. Artificial Intelligence, **20**:215–251.

Clancey, W.J. (1985) Heuristic classification. *Artificial Intelligence,* **27**:289–350.

Davis, R. (1976) *Applications of meta level knowledge to the construction, maintenance, and use of large knowledge bases.* Ph.D. thesis, Stanford University. Report STAN-CS-76-564.

Eshelman, L., Ehret, D., McDermott, J., and Tan, M. (1987) MOLE: A tenacious knowledge-acquisition tool. *International Journal of Man–Machine Studies,* **26**:41–54.

Feigenbaum, E.A. (1977) The art of artificial intelligence: I. Themes and case studies of knowledge engineering, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* Cambridge, Massachusetts, pp. 1014–1029.

Friedland, P.E. and Iwasaki, Y. (1985) The concept and implementation of skeletal plans. *Journal of Automated Reasoning,* **1**:161–208.

Gale, W.A. (1987) Knowledge-based knowledge acquisition for a statistical consulting system. *International Journal of Man–Machine Studies,* **26**:55–64.

Hannan, J.J. and Politakis, P. (1985) ESSA: An approach to acquiring decision rules for diagnostic expert systems, in *Proceedings of the Second Conference on Artificial Intelligence Applications,* IEEE Computer Society Press, Miami, Florida, pp. 520–525.

Kahn, G., Nowlan, S., and McDermott, J. (1985) Strategies for knowledge acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **7**:511–522.

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

Klinker, G., Boyd, C., Genetet, S., and McDermott, J. (1987) A KNACK for knowledge acquisition, in *Proceedings AAAI-87,* American Association for Artificial Intelligence, Seattle, Washington, pp. 488–493.

Marcus, S., McDermott, J., and Wang, T. (1985) Knowledge acquisition for constructive systems, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, California, pp. 637–639.

McDermott, J. (1986) Making expert systems explicit, in *Information Processing 86,* International Federation of Information Processing, Dublin, Ireland, 1986, pp. 539–544.

Musen, M.A. (1988) *Generation of Model-Based Knowledge-Acquisition Tools for Clinical-Trial Advice Systems.* Ph.D. thesis, Stanford University. Report STAN-CS-88-1194.

Musen, M.A., Fagan, L.M., Combs, D.M., and Shortliffe, E.H. (1987) Use of a domain model to drive an interactive knowledge-editing tool. *International Journal of Man–Machine Studies,* **26**:105–121.

Newell A. (1982) The knowledge level. *Artificial Intelligence,* **18**:87–127.

Regoczei, S. and Plantinga, E.P.O. (1987) Creating the domain of discourse: ontology and inventory. *International Journal of Man–Machine Studies,* **27**:235–250.

Shaw, M.L.G. and Gaines, B.R. (1987) KITTEN: Knowledge initiation and transfer tools for experts and novices. *International Journal of Man–Machine Studies,* **27**:251–280.

Shortliffe, E.H., Scott, A.C., Bischoff, M.B., van Melle, W., and Jacobs, C.D. (1981) ONCOCIN: An expert system for oncology protocol management, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* Vancouver, British Columbia, Canada, pp. 876–881.

Tuhrim, S., Reggia, J.A., and Floor, M. (1988) Expert system development: Letting the domain specialist directly author knowledge bases, in Hendler, J. (ed.) *Expert Systems: The User Interface,* Norwood, New Jersey: Ablex, pp. 37–56.

Winograd, T. and Flores, F. (1986) *Understanding Computers and Cognition: A New Foundation for Design,* Norwood, New Jersey: Ablex.

Paper presented at the 2nd European Knowledge Acquisition for Knowledge Based Systems Workshop, Bonn, June 19-23, 1988.

Knowledge Acquisition for Intelligent Tutoring Systems

Jacobijn Sandberg

Radboud Winkels

Joost Breuker

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

Abstract

This paper presents a major technique for knowledge acquisition: The Wizard of Oz technique. Essential is that some expert is asked to simulate the functions of a future system. The data thus gathered are of such a complex nature that they cannot be analysed without the use of an interpretation framework that forms a partial theory on the processes under study. This combination of empirical research and theory development is illustrated by the presentation of research being done in the EUROHELP project.

1. Introduction.

Compared to the freedom children have gained in these modern days the traditional Computer Assisted Instruction (CAI) programs have a very old-fashioned outlook on life (cf. Barnard & Sandberg, 1988). They deny their pupils the right of taking initiative. All initiative is taken by the system, all the major decisions concerning the curriculum are taken by the system. The pupil is restrained in spontaneous behaviour, not because spontaneousness is bad in itself, but its inherent flexibility and seemingly unpredictability is something the system cannot handle. Of course, the rigid character of CAI-programs as it is sketched here, is slightly exaggerated. Nevertheless, other objections have been raised, related to the incapability of such programs to diagnose individuals knowledge-levels and skills at a level detailed enough to be of consequence for the structuring and presentation of the material to be taught. Because of this the programs teach in a rather superficial manner. They are able to detect some student's input as an error, but lack the means to infer what caused the error. They only are able to provide feedback that stated the "right answer" without an accompanying explanation. What educational researchers aim for were systems of quite a different scope. They long for educational systems that are able to develop a model of the knowledge and misconceptions of a student in the course of interaction with the student. Such a model, preferably a runnable model should indicate what a particular student knows and does not know and should thus support deciding what subject matter still needed to be

taught. To be able to teach, a model of a student's knowledge and misconceptions is not enough. A model of the intended knowledge, the knowledge to be acquired by the student is also needed (expert model). The expert model describes the subject matter in a structured way and reflects expertise in a certain domain. However, what is still lacking is teaching expertise. So, we should have a model reflecting teaching skill (some teachers do better than others even if their subject-matter knowledge is equivalent). Now we "know" what the student knows, we "know" what expertise we would like him to acquire, and we "know" how to transfer the knowledge. However, still there is something missing. How is the student supposed to interact with the system? We need a model of interaction as well, reflecting knowledge on processes of communication between system and pupil.

At this point two sides of one and the same coin meet: on the one side education progressing in the direction of individualisation, on the other side techniques enabling computer instruction to be truly individualised as well.

2. Knowledge Elicitation Techniques for educational systems.

In order to build a instructional system, whether it be a traditional one, or an "intelligent" one, a lot of knowledge has to be charted. However, in the latter case, this process of knowledge acquisition for the development of a future system calls for the use of different techniques. The main difference between traditional and intelligent Computer Assisted Instruction is that the four above presented types of expertise have to be represented explicitly in the latter case. Considering humans are the best teachers we know it seems natural to try and capture their knowledge. The next picture shows the four components proposed for Intelligent Tutoring Systems. The table placed beyond the picture lists the main techniques that are used in eliciting knowledge for the development of Intelligent Tutoring Systems.

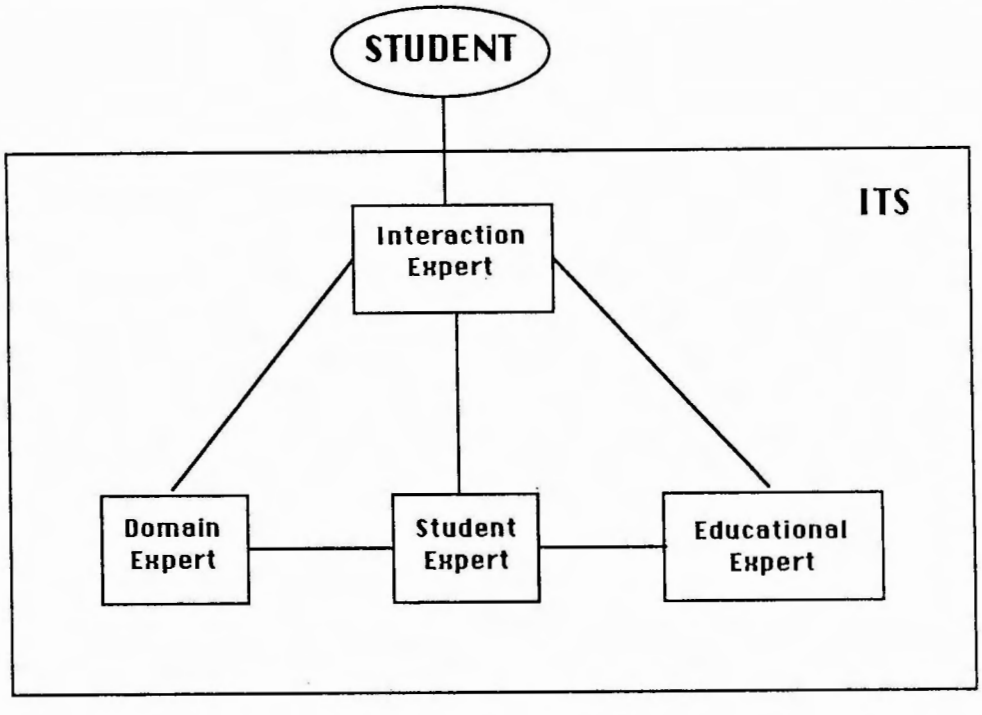


Table 1: Knowledge Elicitation Techniques

Domain Expert	**Student Model**	**Teaching Expert**	**Interaction Expert**
Interview	Interview	Interview	
Introspection	Introspection	Introspection	
Thinking-aloud	Thinking-aloud	Thinking-aloud	
Scaling/Sorting	Scaling/sorting		
Review	Review	Review	
Field Study	Field Study	Field Study	Field Study
Mock-up Study	Mock-up Study	Mock-up Study	Mock-up Study

The techniques listed in the table are the following:

1) **Interview**: Interviews may take many different forms, from totally unstructured to well structured in advance. One of the main problems with interviews is that often the sought for information remains hidden. Interviewing an expert in some field will partially reveal how the expert has his or her knowledge structured, but will often not reveal in a detailed manner how this knowledge is used in problem solving. Generally, experts are not unwilling to provide the latter kind of information; they are just incapable of doing so. The procedures they use in problem solving are often highly automated and therefore not open to direct inspection. The data gathered in interviews are of many sorts and not easy to analyse. Transcribing and analysing interviews is a time consuming and costly affair.

2) **Thinking-aloud procedure**: Asking someone to think aloud during problem solving or task-performance is a very suitable technique to gather process-data. However, not all tasks are suitable for this technique. Some tasks place such a high load on short-term memory that simultaneously having to talk is out of the question. Other tasks imply materials that have to be recoded in semantic terms before they can be talked about (cf Ericsson & Simon, 1984). This process of recoding may distort the original processing and thus result in less valid data. In case problem solving by children is under study, it is not advisable to ask them to think aloud. Many researchers have reported that children under twelve are unable to think aloud during problem solving. An alternative in this case is asking a child to play teacher and explain to the experimenter how some problem has to be solved. Transcribing thinking-aloud protocols and subsequently analysing them is very costly.

3) **Scaling and sorting**: Scaling and sorting is often used to discover relations between concepts. For instance, asking an expert to sort problems on the basis of their difficulty, or on the basis of the solving-procedures attached to different kind of problems. This technique is not only used to investigate expert-knowledge, but also to investigate the knowledge-structures of beginners in some domain. The technique is fairly easy to use and does not cost too much time and effort.

4) **Introspection**: Having someone inspect and report inner thoughts and processing. This technique is not advisable. The data gathered tend to be hardly reliable. There is no guarantee that what is being told really reflects the process that is the target

of the introspection (see Ericsson & Simon, 1984).

5) **Review**: Often reviews are used to clarify ambiguous statements or silences in thinking-aloud protocols. Reviewing is always after the fact. For instance, asking an expert how he solved a previous problem. Reviewing has always the risk that what is being told reflects how some process is being remembered or interpreted instead of how the process actually took place.

6) **Field study**: Data on information processing can also be gathered in the field. Observing how teachers interact with pupils in a classroom, observing how college-professors give lectures, etc. These kind of data are always difficult to interpret. The situation can not be experimentally controled and therefore systematic data gathering is not possible. In most cases, these kind of data are used in an exploratory manner. They provide the researcher with hypotheses that have to be investigated systematically.

7) **Mock-up study (Wizard of Oz)**: Doing a mock-up study implies simulating some future system and its environment (cf. Bierman & Kamsteeg, 1987). For instance, having an expert play the part of a future Intelligent Tutoring System. This technique preserves the realistic flavour of a field-study, but fortunately lacks the disadvantages of the latter. The situation that is being created in a mock-up study is much more controllable than situations in which field studies take place. An expert is asked to perform the functions of a future interactive system by having him communicate with someone (user or student) via a terminal-screen and keyboard. Having both participants communicate by typing has the advantage that they can be asked to think aloud as well. Moreover, the forced communication via a terminal has the advantage that it slows down communication speed and may thus reveal otherwise unnoticed steps in the communication process.

In a Mock-up study, one not only gathers data on how the interaction between user and "simulated system" proceeds, but also data (thinking-aloud, and user-performance) that are relevant to the other components of a future Intelligent Tutoring System. The thinking-aloud protocol of the tutor will shed light on didactic considerations, why are concepts taught at some particular stage, and why are they explained in a particular manner. Also, the remarks the expert makes will reveal something of the nature and structure of expertise and will thus provide additional information on the expert-component. The thinking-aloud protocol of the user will partly reveal what the user knows and what misconceptions he or she has, and therefore will provide data usefull for user modelling. As follows, this technique is the only one that provides reliable data that bear on all four components. And therefore, we feel that this technique is crucial to the development of Intelligent Tutoring Systems. However we do not mean to say that this technique alone is sufficient. It always has to be supplemented by some of the other techniques.

Thus far, we shortly described the techniques that are used in knowledge elicitation for intelligent Tutoring Systems without specifying how the data that are gathered have to be analysed. As we feel the *Wizard of Oz* technique to be the most important one to the development of Intelligent Tutoring Systems we like to explain this technique in greater detail in the next paragraphs. In combination with discussing the technique, data-analysis will be reviewed.

3. The Wizard of Oz technique.

In order to gather data on the possible performance of a future system, one would ideally try to create a situation that approximates the future system as closely as possible. An expert is asked to perform the functions of a future interactive system by having him communicate with a user via a terminal-screen and keyboard. It is evident that humans bring some capabilities along that a future system will never have. To ensure that the future situation is approximated as closely as possible, and thus maximally valid data are gathered, the experimental set-up has to constrain the human expert who acts as a computer in some respects, and to extend his capabilities in others. The next figure presents information on the important differences between humans and automated systems and lists the remedies that may minimize these differences, in the sense that humans better mimic systems.

HUMANS	AUTOMATED SYSTEMS	REMEDIES
knowledge of semantics of natural language	no or limited knowledge of semantics of natural language	hiding content
flexible communication skills	less flexible communication	different rooms low band width communication
natural language preference	multi-media available	multi-media interface accessible for expert
limited memory	less limited memory	command history interaction history

Fig. 1: Important differences between humans and systems

The first listed difference, language understanding is important, because imagine for instance a human coach guiding a novice in working with some mail-system. This coach may well be aware of the meaning of messages being sent and received and may therefore partially base his understanding of what the user is doing on message interpretation. However, a HELP-system performing the same coaching task would never be aware of the meaning of messages, and therefore it is advisable to make sure that a human participating in an experiment is equally incapable of message interpretation. This hiding of content may take two forms: scrambling, or filtering. The second difference concerns communication-skills. In communicating with one another humans use all sorts of means a computer system lacks: gestures, pointing, intonation, facial expression. The third difference illustrates the fact that the envisioned future interface may provide facilities human experts do not have, such as 'direct manipulation facilities' (Hutchins et al., 1985) In that case the future interface already has to be available to conduct the experiment. And, finally, human memory-span is restricted in comparison to that of a computer-system and therefore has to be extended by some external device, a history of what has been going on.

Therefore, the experimental set-up prescribes that both participants are placed in separate rooms and are only allowed to communicate via a terminal. The separation also has the important advantage that both participants may be asked to think-aloud.

3.1. Interpretation framework.

Interpreting empirical data always calls for an interpretation framework (Ericsson & Simon, 1984). Otherwise one cannot see the wood for the trees. In general, a distinction is made between 'weak' and 'strong' frameworks. A 'weak' framework provides a means to categorize data, without structuring or relating the specified categories. A 'strong' framework specifies structuring relations between categories. The research objectives one has predetermines not only the type of data that have to be gathered but also guides the selection of an interpretation framework. A general research objective for example may be to construct a model of a cognitive process. In analysing data on such a process, the interpretation framework has to provide a (partial) theory on what this process looks like and how it reveals itself in the data. A more specific research objective may be to establish the effect of a variable on an other variable. It is obvious that in the former case a more complex, detailed framework is required and in particular that this framework be an explicit one.

3.2. An example: EUROHELP.

The next paragraph illustrates the research technique, explained above, as it has been applied in the EUROHELP-project. This project is aimed at the development of a Shell for Intelligent HELP-systems and investigates what domain-independent knowledge such systems contain. A Help System supports users of Information Processing Systems (i.e. interactive computer programs) in handling and mastering such systems. It not only answers questions, but also 'looks over their shoulders' and interrupts when appropriate.

In order to gather data on the possible interaction between the user and a future Help-system, the Wizard-of-Oz technique has been applied in several experiments (e.g. Bison & v.d. Pal, 1985; Winkels e.a., 1986).
In one of these experiments, the Unix-Vi-editor has been chosen as a domain (Winkels e.a., 1986). The experiment is conducted as follows. A user and a tutor are seated in two separate rooms. Both have two terminals at their disposal, one terminal is used for processing Vi, the other for the interaction between the user and the tutor. The user is working with Vi, following specific instructions containing tasks to be performed. The tasks chosen are representative for text-editing in general: inserting text, changing text, moving text, replacing text, copying text, deleting text, etc. The tutor sees the text the user is editing on one of his terminal screens. However he sees the original text in a scrambled form, which means that all characters are systematically replaced by others. This transformation is meant to ensure that the tutor does not deduce the intentions of the user by relying on semantic textual cues. To overcome the shortcomings of human memory, the last 20 commands issued by the user are listed on the status line, so the tutor's memory is extended by visual means. Seven (naive) users participated in the experiment. Fig. 2 presents a schematic view on the experimental set-up.

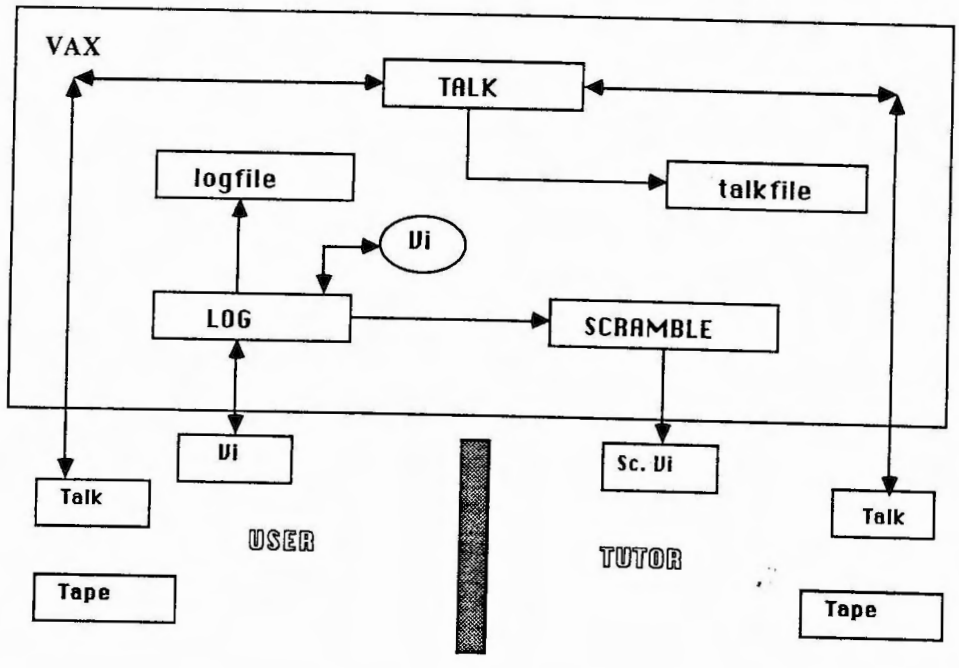


Fig. 2: Simplified scheme of the experimental setup.

Four types of data were collected in the experiment:

1) logfile: record of the ongoing Vi-session, replayable in real time
2) talk file: record of the ongoing dialogue between user and tutor extended with time-stamps. These time-stamps are added so talk file data may be related to logfile data.
3) thinking-aloud protocol of the tutor
4) thinking-aloud protocol of the user

Fig. 3 (see next page) illustrates the types of data gathered. The first column shows the time-stamps. The second column shows the sequence of commands issued by the user during that episode. The third column presents the communication data. The fourth column presents part of the Thinking-aloud protocol of the tutor. The fifth column presents the same data, but in this case for the user.

LOGFILE		TALKFILE	TAPE	TAPE
Time	**Cmnd**	Tutor/User dialogue	Think-aloud Tutor	Think-aloud User
			What is he doing now...	
2050	**i**		insert...	Let's see...
2051	**[sp]**			now a return to get a
2105	**^M**		new line...	new line... Aie...
2115	**u**		u... undo	Well, an 'u'...
			Actually it is a terrible editor for beginners.	
		"tutor" "2120" "18"		
		As a consequence of the i command you gave, you are back in insert mode", that means all you type in will actually be inserted in the text.		Oh, we first have to leave...
			Will he be able to undo it?	
2190	**^[**			
2197	**x**			
2205	**u**			He, why doesn't it...
2214	**^[**		Should I teach him Join?	Now I will first have
2319	**^[**		If he has to get the text back...	to remove that u, so... x...
2324	**x**	"tutor" "2294" "26"	Let's assume he has to...	
		You cannot wipe out a new line (return), by delete or x. But you can join two lines again into one line by moving the cursor to the upper line and then giving the J command. Thus try an arrow up and J.		
2332	**^K**			
2344	**j**		He used the small j.	
		"tutor" "2352" "9"		
		It should be capital J.		
2362	**^K**			
2366	**J**		Yes! I'll tell him...	
		"tutor" "2457" "2"		
		Excellent.		

Fig. 3: An example of the data gathered.

3.2.1. The interpretation framework in EUROHELP.

The research objective in conducting these experiments is the construction of a model for providing help. For the moment we are not concerned with the identification of user problems nor their interpretation, we aim for a model of providing help, given some description of a user problem.

The interpretation framework developed thus far consists of a three-layered architecture (Winkels et al., 1986, Breuker, 1988), similar to the one proposed by Woolf and McDonald (1984):

DIDACTIC GOAL GENERATOR. Didactic goals are overlays of domain concepts, but they provide a didactic view of the domain; they form the hidden curriculum of a HELP-system and are similar to 'genetic graphs' (Goldstein, 1982). Like a genetic graph, didactic goals are related. The interrelations specify the didactic principle that underlies learning a new concept given the fact that another concept is already mastered. Didactic relations are based on principles of (machine) learning, and specify 'least effort' knowledge and skill acquisition trajectories through the domain representation.

STRATEGY PLANNER. The second layer contains a planner that constructs coaching strategies. This planner needs a formal description of a current user problem as input. Its output is a plan, consisting of strategies and substrategies, ending in a sequence of actions (tactics) to be undertaken to meet the user's problem.

TACTICS. The third layer contains the tactics, and is a data structure. Tactics are the terminal elements of strategies, They are the 'speech acts' uttered by the HELP-system.

3.2.2. Data analysis.

The first part of the data-analysis was solely concerned with the identification of "tactics". The main source of data for this analysis were the talk-files containing all tutor actions. To code the protocols in terms of tactics all protocols were partitioned into episodes, which are assumed to correspond with strategies, and each episode into elementary statements. Following this partitioning, each and every proposition has been coded as a particular tactic. The types of tactics were derived from the model description.

Coding the talk-files revealed that the originally proposed typology of tactics was both incomplete and insufficiently fine grained. This was evidenced by the fact that sometimes a statement did not reflect any of the proposed tactics and sometimes statements clearly serving different purposes had to go by the same name. This outcome led to the revision and extension of the originally proposed typology.

Apart from coding the statements in terms of tactics, we looked for recurring clusters of tactics. For, a particular sequencing of tactics that recurs in similar situations may denote a strategy. Indeed, frequently recurring clusters were found. reflecting strategies such as "remedials" and "expansions". A remedial is used in case an error has been made. An expansion is used when it is possible to expand the users knowledge by linking new information to known concepts.

The thinking-aloud protocols of the tutors contained additional information on the importance and the nature of planning in tutoring. The protocols provided evidence for a two-stage model; first identification and interpretation of a users current problem, and secondly planning how to transfer the information needed to overcome the problem.

3.2.3. Implementing part of the interpretation framework.

A prototype strategy planner (second layer of the interpretation model) was implemented in Prolog (cf. Winkels, 1987). This implementation was partly based on the specification of the three layers presented in the previous section, and partly on a sample of communication protocols from the experiment. The construction of this planner is described in detail in Breuker et al. (1987). The implemented Coach is able to plan sequence of actions when fed with a formal description of a users current problem (local need). This formal description specifies the immediate cause that led to the identification of the problem and a diagnosis (lack of knowledge or misconception), and a topic: what the lack of knowledge or the misconception is about. The output of the planning process is a tactic-structure (cluster of different tactics). The Coach developed in the EUROHELP-project is based on the notion of *skeletal planning* (Friedland & Iwasaki, 1985). In reaction to a current problem, the Coach first tries to find a strategy to meet the problem in a library of stereotypical ones. If it does find one, the strategy is instantiated in the current situation and the planning process halts. If no stereotypical plan is available, general, "fall-back" plans are used. These require real planning through the application of refinement rules. The planner reflects a prescriptive view on providing Help and takes different rational principles on how to effectively present information into account.

3.2.4. Evaluation of the implemented model.

In order to compare the strategy planner (PSP) and the empirical data, current user problems were inferred from a new set of empirical data, not used in model construction. These user problems were formalised in terms of 'local needs' and subsequently fed to PSP. The output of PSP was then compared to the original human tutor output in reaction to the same local needs.

The comparison showed that there is a remarkable agreement between the human tutors and PSP. However, in evaluating one is not just interested in being confirmed in ones ideas, discrepancies are also of concern. This citation reflects the general importance of negative information, capable of falsifying some notion or theory. Discrepancies between the data and PSP are therefore of more interest to us than agreements. Whereas, in general, discrepancy between data and theory would automatically imply refuting the proposed theory, this need not necessarily hold in our case. We do not aim at the construction of a model that simulates a human tutor, but we aim at a prescriptive model for providing help. Human tutors not necessarily provide 'optimal' help at all times, and therefore all discrepancies between tutor output and PSP have to be carefully analysed. Studying the discrepancies between PSP output and that of our human tutors revealed that that PSP often generates more suitable help than the human tutors. Human tutors are often presenting information in a manner that violates the rational principles for effective transfer of information. They, for instance, often present too much information in one go, which is shown by the fact that users do not pay any attention to the extra information and therefore do not remember it.

Besides comparing the tactics used by PSP to those of humans, the (frequency of) recurring tactic clusters generated by PSP have been compared to the (frequency of) tactic clusters in the communication protocols. It appeared that most recurring tutor clusters were generated by PSP as well.
The first example denotes pure help. The user is guided but receives no explanation of the new information, and therefore this information is expected not to be remembered. The second example is a pure teaching cluster. A *direction* is given to

Table 1: Examples of recurring clusters

HELP	interrupt
	instantiation
	limitation
	direction
TEACHING	interrupt
	instantiation
	direction
	description
	operationalisation

explain how to reach a goal in operational terms. However, the concepts used in the *direction* are explained with the subsequent description. The *operationalisation* provides additional operational information and explains the *direction* in greater detail. Comparing the frequency of occurrence of the recurring clusters revealed some differences. The communication protocols contain a lot of clusters that represent **pure help**. PSP is more directed at linking new information to prior knowledge. Partly, this difference reflects the difficulty of the task the human tutors had. They were expected to respond quickly to user needs, and therefore they sometimes tended to be too concise. PSP on the other hand was concise as well, but whenever new information could be linked to prior knowledge, this link was explicitly stated. In such a case, new information would be presented in terms of a concept the user already knew. For example, if a user knew the i-command which gets you into insert mode in Vi, and was presented with a new insert-command, PSP would say: "X (new command) does the same as Y, but ..". Human tutors were more inclined to just say: "X does .." without referring to a concept already known.

4. Concluding remarks.

The knowledge Intelligent Tutoring Systems need in order to be able to interact in a flexible manner with a student has to be attained. Several techniques are available to acquire this knowledge; one of these techniques, the Wizard of Oz technique shows great promise. Applying this technique provides a lot of data needed for all four components of an Intelligent Tutoring System. As has been described, just applying the technique is not enough. The data gathered in this manner have to be analysed. And, because the processes under study are of a fairly complex nature, analysing the data calls for an interpretation framework that structures and directs the analysis.
What does this sum up to? The combination of data gathering and model construction as outlined in this paper is a first step towards the development of a methodology for Intelligent Tutoring Systems. In analogy to the development of a methodology for building Expert Systems (Breuker et al., 1988) we aim for a methodology for building Intelligent Tutoring Systems. The Wizard of Oz technique" should evidently be part of such a methodology, but is not a complete methodology in its self. Such a methodology would not only describe what techniques for knowledge acquisition are available, but also provide the developer with prescriptive guidelines on when and how to use what technique. Besides, it should provide a library with interpretation models reflecting

different types of teaching and coaching paradigms. A research project aimed at methodology-construction is planned in the near future.

5. References

Barnard, Y.F., & Sandberg, J.A.C. 1988. Where is the interaction component in Intelligent Tutoring Systems? Paper presented at the annual meeting of the AERA, April, New Orleans, 1988.

Bierman, D.J., & Kamsteeg, P.A. (1987). A Methodology for the Development of 'Intelligent' Educational Systems. In: C. Schwazner & B. Seipp (Eds.) *Trends in European Educational Research*, p. 153-166. Dusseldorf: Bramschweiger Studien Reihe.

Bison, P., & van der Pal, F. (1985). Using UNIX-Mail: An experiment in on-line tutoring. University of Amsterdam (the Netherlands).

Breuker, J.A. (1988). Coaching in Help Systems. In: J.Self (ed.), Intelligent Computer-Aided Instruction. London: Chapman & Hall.

Breuker, J.A., Winkels, R.G.F., & Sandberg, J.A.C. (1987). A shell for Intelligent Help Systems. *Proceedings of the 10th International Joint Conference on Artificial Intelligence, 1*, pp. 167-173.

Breuker, J.A., Wielinga, B.W., van Someren, M., de Hoog, R., Schreiber, G., De Greef, P., Bredeweg, B., Wielemaker, J., Billault, J., Davoodo, M., Hayward, S. (1988). Model-driven Knowledge Acquisition: Interpretation Models. Deliverable task A1, Esprit Project 1098. University of Amsterdam

Ericsson, K.A., & Simon, H.A. (1984). *Protocol Analysis: verbal reports as data.* Cambridge MA: MIT Press.

Friedland, B.E., & Iwasaki, Y. (1985). The concept and implementation of skeletal plans. *Journal of Automated Reasoning, 1*, pp 161-208.

Goldstein, I.P. (1982). The Genetic Graph: a representation for the evolution of procedural knowledge. In: Sleeman, D., & Brown, J.S. (Eds.), *Intelligent Tutoring Systems. New York: Academic Press.*

Hutchins, E.L., Hollan, J.D., & Norman, D.A. (1985). Direct Manipulation Interfaces. San Diego: University of California, ICS Report 8503.

Winkels, R.G.F., Sandberg, J.A.C., Breuker, J.A. (1986). Coaching Strategies and Tactics of IHSs. Deliverable 2.2.2., ESPRIT Project P280 "EUROHELP". University of Amsterdam.

Winkels, R.G.F. (1987). PSP: A Prototype Strategy Planner for IHSs. Deliverable 2.2.4, ESPRIT Project P280 "EUROHELP". University of Amsterdam.

Woolf, B., & McDonald, D.D. (1984). Context dependent transitions in tutoring discourse. *Proceedings of AAAI 1984.* Los Altos: Kaufmann.

GENERATION OF RULES WITH CERTAINTY AND CONFIDENCE FACTORS FROM INCOMPLETE AND INCOHERENT LEARNING BASES

M. SEBAG([1])([2]), M. SCHOENAUER([1])

Abstract : A "learning rules from examples" system is presented. The learning base is made indistincly of examples or rules, giving the actual conclusions corresponding to known entry configurations. The entries are multi-valued, and the conclusions boolean.
A notion of weak coherence is defined. Under this hypothesis, the reduction step produces an equivalent set of rules, of optimal conciseness.
The extrapolation step then extends these rules on the unknown parts of the learning domain, in the limits imposed by the coherence with the initial base.
The quality of learning is then evaluated : the generalized rules are attached a confidence factor, measuring the part of unknown they cover, and a certainty factor, taking into account the initial incoherence of the base.
The first results of the algorithms are presented.

([1]) Laboratoire de Mathématiques Appliquées, Ecole Polytechnique, 91128 PALAISEAU Cedex.
([2]) Laboratoire de Mécanique des Solides, Ecole Polytechnique, 91128 PALAISEAU Cedex.

INTRODUCTION

The development of industrial expert systems naturally sets the problem of knowledge acquisition, asking the following questions :

When experts are unable to state the rules (as in pattern recognition) [Lauriere 86], are expert systems to be abandonned ?
What kind of technique could help knowledge acquisition, lowering its cost ?
Whenever knowledge is written out, is there a way to validate it (is it reasonably true) and estimate it (is it trustworthy) ?

Partial answers to these questions can be brought by machine learning.
The main trends of machine learning nowadays can be set as learning pertinent concepts in a given domain [Lenat 76], [Diday 76], optimizing the solution of a well-posed problem [Michalski 83] or summing up a well-known problem [Quinlan 84].

We deal here with a *sufficiently known* domain : we suppose pertinent conclusions of the system are identified, as well as sufficient symptoms, or entries.
The learning base is made of experiments, which can be of any level of generality (examples and counter-examples as well as rules). Each experiment gives the state of (some of) the N entries, and the corresponding state of the S conclusions to these entries.

We make the following hypotheses :
Entries and conclusions are distinct ;
Entries are nominal variables ;
Conclusions can take boolean values.

The learning universe is made of any possible configuration of the entries, together with the corresponding conclusions of the system ; so only a (usually small) part of this universe is actually known and represented in the learning base.

In formulating our problem (in Part 1), we had to deal with *coherence*, and to define the **weak coherence** of such a base. As we first worked on industrial maintenance and diagnosis problems, we had to suppose *a priori knowledge* on the domain (default conclusions, authorized exceptions ...), which implied conflict resolution rules. This can easily be adapted to any other domain where a priori knowledge exists (non destructive control, signal analysis of a known phenomenon, ...).

After coming to a coherent base, we transform it with two types of **generalization.**

In Part 2, we proceed to the **reduction** (or generalization by deduction) of the base, generating necessary and sufficient rules equivalent to the initial information. This step also detects and treats redundancy of the base, as well as incomplete generalizations.

The aim of this reduction step is very similar to that of the PLA optimization (Programmable Logical Arrays). Given a truth table of boolean conclusions, these have to be realized as a AND and OR matrix of the entries. The objective is to reduce the length of this matrix, to optimize its physical implementation [Hong 74], [Sebag 85]. A rule-based implementation of PLA optimisation is given in [Saucier 87].

In Part 3, relaxing the completeness hypothesis, we **extrapolate** (generalization by induction [Mitchell 82]) the reduced rules to the unknown entry configurations. Our aim is there to cover the largest possible part of the learning domain, preserving the coherence with the counter-examples of the initial base.

Extrapolated rules are "simple", and therefore can be easily criticized by experts, who can then add to the learning base any limiting counter-example.

But the part of unknown in each extrapolated rule has to be quantified for such a base to be of any practical use. This is done after defining a **relative measure** on the learning lattice (Part 4). We are then able to qualify the quality of the learning by attaching a **confidence factor** to every rule : knowing is not enough, we must know what we know.
Using this measure, we are also able to relax the coherence hypothesis, reintroducing all the non-coherent examples that were "thrown away" in the first step. We generate a **certainty factor** on the conclusions (probability of the conclusion viewed as asymptotic frequency [Bhatnagar 86]). Moreover, possible repetitions, as well as strong contradictions in the initial base modify this certainty factor.

Our program can be fed with elementary terms (that is, with a unique possible value for every entry) in the case where only such knowledge is available ; it then *automatically generate a set of rules* for further diagnostic, supposing these examples sufficiently cover the domain.

But it can also be used as a *pre-processor for existing rule bases*, eventually completed with examples of unformulated rules, refining and validating such bases.

For more technical details on these algorithms, one can refer to [Sebag 87], which deals with the case of boolean entries, but uses the same basic ideas.

We present the first results in Part V. The algorithms have been programmed using the C langage on IBM/AT compatible computers.
The total computing time, when compared to similar programs, remains small : for instance, an example base of 200 experiments with 30 entry parameters and 2 conclusions leads to about 12 extrapolated rules in about 20 minutes.
The limits we reached took place in the extrapolation step, where the amount of memory we needed was too high, for a base of 500 experiments with 50 entries and 2 conclusions, while the reduction step was achieved for this base in about 27 minutes.

I DEFINITIONS AND NOTATIONS.

We suppose that the learning domain is structured in N **entries**, nominal variables which can take values among a finite set, and S boolean **conclusions**. Entries and conclusions are distinct.
Let q_i be the number of possible values for the ith entry.

I.1 Experiments and rules

Each experiment is defined by N **hypotheses** e_i, which are subsets of $[1,q_i]$ respectively. The q_i possible values of entry i are mutually exclusive, and each hypothesis e_i modelizes something like

Color = Red or Blue or Green,

that is, no matter which value of the subset e_i the ith entry of the experiment takes, all other entries being unchanged, the conclusions of the experiment will be the same.

An hypothesis equal to $[1,q_i]$ will be denoted by X, as the value of this entry is *indifferent* ; it has no influence on the conclusions in the context.

A conclusion (denoted by f_j, for j = 1,..,S) can be True (1), False (0) or missing (?), for instance if it is obscured by some other conclusion.

The rules we shall generate by generalization fit in the same formalization: if some entries are not specified as hypotheses of a rule, it definitely means that, whatever the value of these entries, the conclusions of the rule stand true.

Hence, in the following, we shall denote by **term** indistinctly *experiments* from the initial example base, as well as *rules*, generated from experiments or given in the initial base.
We shall too distinguish **examples** (or j-examples if necessary), terms of conclusion f_j true, from **counter-examples** (or j-counter-examples) terms of conclusion f_j false.
Finally, an **elementary case** will be a term in which every entry is uniquely specified, i.e. $\#(e_i)=1$, for i in [1,N].

Remark 1 : This unique formalization for experiments and rules implicitly suppose that every experiment is reproducible. This will in fact be stated more precisely in Part 4.
Moreover, it allows to handle existing rule bases in order to validate and refine them.

I.2 Learning basis.

The learning base is made up of P terms, each one of them characterized by its N hypotheses, the possible values of its N entries, and by the corresponding S conclusions.

$$T \quad \Leftrightarrow \quad e_i(T)_{i=1..N},\ f_j(T)_{j=1..S}$$

with $e_i(T) \subset [1, q_i] \Leftrightarrow$ in term T, the ith entry takes its value indifferently in $e_i(T)$.

$e_i(T) = X \Leftrightarrow$ the value of the ith entry has no effect on the conclusion of term T.

Définition 1 : *We define the* **precision** *of a term* T *by*

$$h(T) = \sum_{i=1..N} \left[\frac{q_i - \#(e_i(T))}{q_i - 1} \right]$$

where $\#(e_i(T))$ *is the number of elements of the subset* $e_i(T)$.

In the boolean case, h(T) is (half of) the number of non-indifferent entries of T.

Définition 2 : *A term* T_1 *is* **covered** *by an term* T_2 *if*

$$\forall\, i \in [1,N], \qquad T_1(e_i) \subset T_2(e_i)$$

T_1 is then an instance of T_2, and $h(T_1) \geq h(T_2)$.

Définition 3 : *A term* T_1 *is* **k-adjacent** *(respectively* **strictly k-adjacent***) to a term* T_2, *for* $k \in [1,N]$ *if*

$$e_k(T_1) \not\supset e_k(T_2)$$

and

$$(\forall i \neq k) \qquad e_i(T_1) \subset e_i(T_2)$$

(respectively, $(\forall i \neq k) \qquad e_i(T_1) = e_i(T_2)$ *).*

T_1 and T_2 are only discriminated by e_k on entry.

I.3 Inheritance and coherence.

Let us consider a specific case, covered by another more general term, and their respective conclusions.

If they have the same conclusion, the specific case is an instance of the general one, and both terms are redundant.
This specific case will be eliminated from the base for both the reduction and the extrapolation steps. But it will be taken into account during the generation of the certainty factor in Part 4.

If they have different conclusions, we have to resolve conflicts, using *a priori* knowledge on the learning domain.

The first applications we worked on were industrial maintenance and diagnosis ; inherited from this context, we suppose the following *a priori* information about the learning domain :

- We are interested in rules concluding to problems (Defect f_j True).
  By default, there is no problem, (f_j *is false).*

- If a conclusion f_j is false in a general case, it can nevertheless be true in a specific case.
  Failure is a specific case, compared to working state.

- But if f_j is true in a general case, f_j must be true for every more specific instance.
  But working state cannot be a specific case of failure.

These common-sense assumptions then allow us to define inheritance laws on the learning lattice.

Définition 4 : *The learning base is* **weakly coherent** *if*

1. A term inherits its conclusions from any covering example ; there is no exception to conclusions f_j *true.*

2. A term which is not covered by any example inherits from the covering counter-examples ; but terms of conclusion f_j *false can have exceptions.*

Remark 2 : Of course, domains with a true default conclusion could be handled the same way. As can any domain where exception is only allowed one-way. But we don't handle domains with exceptions to exceptions.

When dealing with a learning base, we shall possibly modify some conclusions, following above rules. These incoherent experiments will be reintroduced in the base to generate the certainty factor on the conclusions (see Part 4).

From now on, we suppose our learning base to be weakly coherent according to definition 4.

From the weak coherence rules we derive

Proposition 1 : *Given an example* Ex_1 *and a counter-example* Ex_2 *of a weakly coherent learning base, there exists* $i \in [1,N]$ *such that one of the following is true :*

i) $e_i(Ex_1) \subset e_i(Ex_2)$ (compatible terms)

ii) $e_i(Ex_1) \cap e_i(Ex_2) = \emptyset$ (discriminated terms)

II REDUCTION.

If the learning base is assumed weakly coherent and complete, that is, covering any elementary case, the aim of the reduction step can be viewed in terms of optimal conciseness :

Définition 5 : *Given a complete, weakly coherent learning base* B, *the* **reduction** *produces a minimal set* R(B) *of necessary and sufficient rules, strictly equivalent to the examples of B :*
For any j in [1,S], an elementary case covered by a term in B *of conclusion* f_j *true, is covered by at least one rule of* R(B) *concluding to* f_j *; reciprocally, an elementary case covered by a rule concluding to* f_j *is covered by at least one example of* f_j.

The optimization criterion of reduction is first, the number of rules in R(B) and second, the sum of their precisions.

In a first step, any possible simplification is considered, and virtual terms are generated, until saturation. The explosive exploration can be limited by considerations about simplifications and redundancy.

Then a minimal set of necessary and sufficient rules is iteratively determined.

II.1 Factorization.

Let T_1 and T_2 be two k-adjacent terms for an entry k, such that T_1 and T_2 concludes to a same conclusion f_j. For instance :

T_1 *Flame* ⇒ *Fire*

T_2 *No Flame AND Smoke* ⇒ *Fire.*

Then the T_2 hypothesis relative to entry e_k can be relaxed, the conclusion f_j still being valid, which gives the term :

T_3 *Smoke* ⇒ *Fire*

This example of generalization is formalized by

Proposition 2 : *Given two k-adjacent terms* T_1 *and* T_2, *both of them concluding to a same conclusion* f_j.

T_1 *can be generalized relatively to* e_k, *with respect to context* T_2 :

Let T_3, *concluding to* f_j, *so that*

$$e_k(T_3) = e_k(T_1) \cup e_k(T_2), \text{ and}$$

$$(\forall\, i \in [1,N],\ i \neq k)\quad e_i(T_3) = e_i(T_1).$$

We call T_3 *a* **generalization of** T_1 **with respect to** T_2, *and denote it* $Gen(T_1, T_2)$. *Moreover, generalization* T_3 *satisfies*

$$h(T_3) < h(T_1)$$

and

$$\{T_1, T_3\} \text{ covers } \{T_1, T_2\}$$

The generalized terms are then added to the base (if not already there). We shall denote these new terms **virtual terms.**

Remark 3 : The origin of the virtual term T_3 from terms T_1 and T_2 can be given to the user, if wanted ; the reduced rules are recursively explainable.
Indeed, this possibility of explanation remains a *sine qua non* condition of intelligent learning ([KODRATOFF 86]).

Remark 4 : The coherence might not be preserved by reduction ; even if initial terms are compatible, virtual examples may cover counter-example (see II.2.1).

We proceed to this elementary generalization until saturation. But the exploration is greatly limited by

Proposition 3 : *Let R be a virtual term generalizing a term T in a context C, that is R = Gen(T,C). If R is not covered, then for any context C' covering C we have*

$$R = Gen(T, C) \supset Gen(R, C').$$

Proof : Let e_k be the entry of T that R generalizes.
C' covers C, C and T are k-adjacent, and C' cannot cover T, because it would otherwise cover R, which by hypothesis is not covered : so C' and T are discriminated by at most entry k.
Hence, T can be generalized in the context C', that is on entry k, which generates R.

When we have generated every possible virtual term, we iteratively construct our minimal family of reduced rules, by selecting at each step the most actually covering term. This term is chosen according to the measure on the lattice described in Part 4.

II.2 Algorithm.

II.2.1 Conflict solution

There are possible conflicts, according to the coherence rules of definition 4, when reading the learning base or when generating virtual term (see Remark 4).
These conflicts are solved by forcing the conflicting conclusions to weak coherence, that is, preference is given to false failure being detected over real failure not being detected :

- Conflict on the very same hypotheses (initial base only) :

True / False or ?	gives	True	*Failure is sometimes true.*
? / False	gives	?	*Failure can be unknown.*

- Conflict between a general term covering a specific case :

The very same rules apply, as True in the general case must be true in the specific case (True / False or ? gives True), and if there is a chance of failure in the general case, we cannot be sure there is no failure in the specific case (? / False gives ?).

II.2.2 Creation of the virtual terms.

The terms are reviewed by decreasing order of precision.
Comparison with non dominated terms (see Proposition 3) possibly leads to generalization, according to proposition 2. The virtual terms created are added to the base, preserving its weak coherence. As these terms have less precision, by construction, they will be reviewed in the following step.

II.2.3 Selection of the reduced rule.

First of all, to insure decreasing of the base, we want a candidate rule R to cover at least one term of the initial base still uncovered. This heuristic has been used by many authors (see [Saucier 87] and [Ho Tu Bao 87], for instance).

We are interested in optimizing the *relative generating power* of terms, that is, the part of the total lattice the term to be chosen covers.
The covering factor used to quantify this generating power will be detailed in IV.3. Notice now it distinguishes three different types of covered terms :

- terms already covered by some rule chosen in a previous step, and in fact eliminated from what remains to cover.
- terms that will be eliminated if rule R is chosen and incorporated in the reduced base.
- Finally, when S, number of possible conclusions, is greater than 1, terms that will be partially eliminated if R is chosen. Some of the conclusions of such terms will have to be covered by some other reduced rules, in a further step.

II.2.4 Update of the learning base.

After selecting a rule, and incorporating it in the reduced base, all terms it covers are updated : the matched conclusions (by coherence) of a covered term are put to "?".
The process goes on (return to II.2.3) until every conclusion of every initial term is eliminated.

By construction, at least one reduced rule is fired (given a true conclusion) by every example of the initial base (in fact, by any elementary case covered by an initial example), and none is fired either on counter-examples or on unknown part of the learning domain.

Moreover, the initial base has been virtually completed with any unknown elementary case associated with conclusions f_j false.

In terms of *a priori* knowledge on the domain, this means that the default state is a "no failure" state, or, that every possible failure is present in the initial base : the initial base was supposed to be **complete.**

III EXTRAPOLATION.

In a real problem, the above completeness hypothesis is usually not valid. The **extrapolation step** attempts to generalize the reduced rules by induction over the unknown parts of the learning domain ([Mitchell 82]).

Theoretically, one could extrapolate directly on the initial base, as it is equivalent to the reduced base (See V).

III.1 The problem of extrapolation.

As reduction was the generation of *necessary and sufficient* rules, extrapolation is that of *sufficient* rules, still according to the initial base, of course. And the way to do that will be, too, by relaxing hypotheses of the reduced rules.

Extrapolation must respect the coherence constraints of initial counter-examples ; but nothing is now supposed on the unknown parts of the learning domain. And it is this unknown part we want to cover with extrapolated rules.

We shall see in Part 4 that a confidence factor attached to every extrapolated rule will measure the part of "guess" it contains, that is, the relative parts of known and unknown experiments it covers.

But let us consider the following example learning base :

E *Color = Grey AND Height = Big ⇒ Elephant*

C *Color = Grey AND Height = Small ⇒ Mouse (NO Elephant)*

The two possible extrapolated rules from example E are

E_1 *Color = Grey ⇒ Elephant*

E_2 *Height = Big ⇒ Elephant*

The extrapolated rule E_1 covers counter-example C, with a contradictory conclusion : it is therefore impossible, acording to coherence rules. The only possible extrapolation of E in this base is E_2.
This small example points out the importance of representative counter-examples in the initial base (these were useless in the reduction step).

III.2 Formalization.

Definition 6 : *Let* R *be a rule of conclusion* f_j *(true). A rule* R′ *is a coherent* **j-extrapolation** *of* R *if*

- R′ *covers* R, *and*
- R′ *is coherent with any initial j-counter-example, i.e.*

$(\exists\ i \in [1,N])$ *such that*

$$e_i(R') \subset_{\neq} e_i(C)$$

or

$$e_i(R') \cap e_i(C) = \emptyset.$$

We shall denote $L_{R'}(R)$ *the subset of [1,N] such that*

$$(\forall\, i \in L_{R'}) \qquad e_i(R) \subset_{\neq} e_i(R').$$

In the following section III.3, we shall not refer to subscript j.

III.3 Coherence of the extrapolation.

A counter-example **C** so generates coherence constraints limiting the possible extrapolations of any rule **R**. Two different situations can arise, depending on wether **C** covers **R** or not (from the coherence hypothesis, **R** cannot cover **C**).

III.3.1 The general case.

In this case, there is no ordering relation between **R** and **C**, according to definition 2.

Let D_C be the set of entries separating **R** and **C** :

$D_C = \{\ i \in [1,N],$ such that $e_i(R) \not\supset e_i(C)\ \}$.

Any rule **R'** extrapolated from **R** must be coherent with **C**, that is

$$\exists\ i \in D_C \text{ such that } e_i(R') \not\supset e_i(C).$$

otherwise, an entry configuration from **C** would fire **R'**, leading to incoherence on the conclusion.

We have stated the following result :

Proposition 4 : *Let* R' *be a rule covering a rule* R. *Then* R' *is a coherent extrapolation of* R *if, for every counter-example* C *not covering* R, *we have*

$$L_{R'} \not\supset D_C$$

Remark 5 : This proposition is the basis of the extrapolation algorithm described in III.4, as it allows the relaxation of non discriminating entries.
A further step (not yet implemented) will be the relaxation of discriminating entries, as in, for instance,

Color = Grey ⇒ *Elephant*
extrapolated in
Color ≠ Red ⇒ *Elephant.*

III.3.2 The specific case.

The counter-example covers here the example-rule : the conclusion is generally false, but is true in the specific case.

We heuristically decided to extrapolate only in the limits of the counter-example : we had in this case the new constraint for extrapolated rules to be dominated by the counter-example. For a detailed discussion of this option, see [Sebag 87].

III.4 Algorithm.

For each j in [1,S], let us consider $\mathbf{B}_R(j)$, the set of reduced rules of conclusion f_j true, and $\mathbf{A}_C(j)$ the set of initial j-counter-examples (initial terms of conclusion f_j false).

Limit cases.

If $\mathbf{A}_C(j)$ is empty, the extrapolation trivially gives that "f_j true" is always true, and part 4 will tell, by the mean of the confidence factor, what can be trusted about such assertion !
If $\mathbf{B}_R(j)$ is empty, we shall as well consider that "f_j false" is always true, as there is no example of failure f_j (back to the default state).

Here again, we shall drop subscript j in the following, keeping in mind the algorithm is to be done for each j in [1,S].

Initialisation.

For every rule in $\mathbf{B}_R$, we define L_R to be the set of degrees of freedom left for the extrapolation of R, and initialize it as

$$L_R = \{ i \in [1,N],\ e_i(R) \neq X \}$$

Constraints generation.

Let R be a rule of $\mathbf{B}_R$. Every counter-example C in $\mathbf{A}_C$ is matched to R, which defines a set $\Delta_R(C)$ of constraints on L_R, or eventually directly reduce L_R, as follows :

If C does not cover R, the coherence rules imply that all discriminating hypotheses cannot be relaxed simultaneously ; the extrapolated rules will so stay separated from C. This gives

$$\Delta_R(C) = \{ i \in [1,N],\ e_i(C) \not\supset e_i(R) \}$$

In the peculiar case where $\Delta_R(C)$ only contains one integer k : the constraints then means extrapolation on hypothesis k is not allowed, and so

$$L_R = L_R \setminus \{k\}$$

If C does cover R, (see III.3.2), then any coherent extrapolation of R must be covered by C, i.e.

$$L_R = L_R \setminus \{ i \in [1,N] ,\ e_i(C) = e_i(R) \}$$

A constraint can sometimes be removed :
- if it is covered by another one, or
- if the set L_R is reduced and no more covers it.

Generation of the extrapolated rules.

From what remains of the set L_R, and the set of constraints
$$\{ \Delta_R (C),\ C \in A_C (f_j) \},$$

we look for maximal subsets $M_k(R)$, if any, such that

$$M_k(R) \subset L_R \quad \text{and} \quad \forall\, C \in A_C (f_j),\ M_k(R) \not\supset \Delta_R(C).$$

This is achieved by classical techniques. The extrapolated rules are then given by

$$\begin{array}{ll} \text{If } i \notin M_k(R), \text{ then} & e_i (R_k) = e_i (R), \\ \text{else,} & e_i (R_k) = X. \end{array}$$

Remark 5 : The extrapolated rules, even without confidence factor, can help the expert : such rules usually are very "simple" (as in the canonical example of extrapolation *All french women are red-haired*), and so, easily refutable. The expert can deny them, and add counter-examples to the learning base to get rid of them.

IV RELATIVE MEASURE ON A RULE LATTICE

We have already mentionned the need for a measure on our rule lattice, in the following situations :

- during the reduction step, in order to choose the "best covering" rule to add to the reduced base ;
- after the extrapolation step, one needs to know, for each extrapolated rule, the confidence he can put in it. Notice that a minimum confidence could as well be imposed during extrapolation ;
- the initial base was forced to be weakly coherent with no repetition. But the information contained in the incoherent or repeated experiments cannot be neglected, and must lead to a certainty factor, depending on the relative amount of examples and counter-examples.

IV.1 An exponential problem.

An absolute measure of some weight of a term is given by

Définition 7 : *The size of a term* T *is the number of elementary terms it covers, i.e.*

$$\text{Card}(R) = \prod_{i=1..N} \#\,(e_i(R)))$$

But our concern here is to estimate a term **relatively to others**, and the numbering of a union of terms (with non empty intersection) is explosive :

- in terms of amount of memory needed, if we just count the terms in the complete set of all possible elementary terms (of size the product of all q_i).
- in terms of computing time, if this set is not available, as the intersections must not be counted more than once.

IV.2 A density approach.

Any of the two above exact approaches being rejected, we tried to evaluate the weight of a specific case relatively to a covering rule in terms of the number and size of terms of equivalent precision. Let us show the basic ideas on the following example :

Example. Consider the following rule R, with a specific example E :

R : *Human* ⇒ *Mortal*
E : *Human AND Town Athenes AND Socrates* ⇒ *Mortal*

We would like to estimate the amount of information brought to R by E, in two situations :

1 Suppose we only have 3 boolean entries, *Human, Town Athenes and Socrates.* **(N=3).**
The size of R is 2^2, and E is an elementary case (of size 1). The density of E relatively to R is equal to its weight, namely here 1/4 (according to common-sense).

2 Suppose now there is another entry *Time 60 AC* **(N=4).** The term E is no longer an elementary case, but is now a sub-lattice. We have

$$Card(E) = 2.$$

The weight of E relatively to R is the number of terms of same precision than E and covered by R, given by

$$C_3^1 * 2^2 = 12$$

And the density of E with respect to R is now 2/12 = 1/6.

We already see on this example that the density of a sub-lattice decreases as the number of entries increases : the relative quantity of information lowers as the number of unknown gets bigger.

We will now write these in the general case.

Définition 8 : *Let* E *be a term covered by a rule* R, *and let*

$$\epsilon_i = \#(e_i(E)) \quad \text{and} \quad \rho_i = \#(e_i(R)).$$

We define the weight of E *with respect to* R *by*

$$\pi_R(E) = \frac{1}{C^{\Sigma(\epsilon_i - 1)}_{\Sigma(\rho_i - 1)} \cdot \prod_{i=1..N} C^{\epsilon_i}_{\rho_i}}$$

$$\pi_R(R) = 1$$

$$\pi_R(\varnothing) = 0$$

and the density of E *with respect to* R *by*

$$\Phi_R(E) = Card(E) * \pi_R(E).$$

In the following, we shall mainly consider, for a rule R, the sum of the relative density of its examples or counter-examples.

Remark 6 : This quantity is skewed. But we are interested in comparing intersecting sets by the ratio of equally skewed quantities.

IV.3 Representativity of a reduced rule.

As we briefly mentionned in part 2, there are three different kinds of terms covered by a rule R to be taken into account :

- the terms whose conclusions are already covered (inactive sub-lattices),
- the terms that will be eliminated if R is chosen as reduced rule (totally dominated sub-lattices), and
- the terms whose conclusions are partially dominated by R (in the case S>1 only). Indeed, the reduction of a learning base with S conclusions is not equivalent to S reductions of bases with one conclusion : considering the conclusions as distinct, and neglecting possible coïncidence of conclusions can lead to generate more reduced rules than initial experiments.

We define a hierarchy on the extended base (initial and virtual terms) : a set Son(R) includes directly covered terms not yet eliminated, excluding "gran-sons".
The representativity of every term is then constructed from sons to fathers.

The choice criterion clearly is the amount of totally dominated sub-lattices, the secondary criterion being the amount of partially dominated sub-lattices, which we formalize by

Définition 9 : *The* **total representativity** *of a term* R *is the sum of the total representativities of its sons multiplied by their relative weights with respect to* R.

The **partial representativity** *of a term* R *is the sum of the total representativities of totally dominated sons, and the total and partial representativities of partially dominated sons, still multiplied by their relative weights with respect to* R.

If R *has no son, its total representativity is its size, while its partial representativity is zero.*

For more details, and in the boolean case, we still refer to [Sebag 87].

These quantities are used respectively as first and second criterion for the choice of the next reduced rule, during the reduction step.

IV.4 Confidence factor for an extrapolated rule.

A confidence factor is attached to every extrapolated rule in order to quantify the unknown part of the learning domain it covers, compared to the known part. Notice that this factor is different from the certainty factor of the conclusion, or probability of this conclusion (here again, the work must be made for each possible conclusion j...) :

If we denote by c_R the confidence factor of a rule, one can say that the conclusion of R are certain in c_R % of the possible cases, but nothing can be said about the remaining cases (unknown cases can have conclusions equal as well as different from those of the rule).

From a semantic point of view, an illustration of this difference can be found in the following two sentences :

I am **almost** *sure this work is over*
I am sure this work is **almost** *over.*

This confidence factor is to be understood and used by a human expert ; therefore, it must not move away from common-sense.

First, let us define the certitude of a rule R by the sum of the densities of sub-lattices of R which are intersections of R with the initial base **B** :

Definition 10 : *For any rule* R, *let*

$$Cert(R) = \sum_{\substack{E \in \mathbf{B} \\ f(E)=f(R)}} \Phi_R (R \cap E)$$

For instance, the certitude and size of a reduced rule are of same magnitude : every elementary case firing the rule is covered by a term in **B**.

But in the case of an extrapolated rule in a base which is small compared to the learning domain (the usual case, so forth), this factor Cert(R) is small and hardly meaningful.

So, instead of examples, let us consider counter-examples : we can apply to counter-examples the same extrapolation algorithm we used to get extrapolated rules. And on parts of the learning domain where indistinctly extrapolated rules and extrapolated counter-examples are fired, we are sure the conclusion is not sure.
We so define the non-certitude of a rule R, NCert(R), using counter-examples of **B** the same way we defined certitude using examples.

Finally, it seems natural to define the confidence factor of a rule R as a function of its certitude and its non-certitude, or, more precisely, using the certitude and non-certitude of every term intersecting R, multiplied by its own confidence factor. This leads to the following adaptative definition of the confidence factor :

Définition 11 : *For any term* R *of* $\mathbf{B}_T$ *(union of the initial base, the extrapolated examples and the extrapolated counter-examples), let*

$$Conf(R) = \frac{Cert(R) + \sum_{R' \in E,\ f(R')=f(R)} Conf(R') \cdot \Phi_R (R \cap R')}{Cert(R) + NCert(R) + \sum_{R' \in E} Conf(R') \cdot \Phi_R (R \cap R')}$$

The existence of a solution to these implicit equations is assured by Brouwer theorem, and the computation is done iteratively, the initial guess fixed to Cert(R) for each Conf(R).

Limit cases

The confidence factor of a reduced rule, under weak coherence hypotheses, is clearly 1 : its intersection with counter-examples, as well as with extrapolated counter-examples, is empty.
Its certitude might be greater than its size if it is over-represented in the initial base.

An extrapolated rule is of confidence 1 if and only if its intersection with extrapolated counter-examples is empty ; its certitude can in that case be used to know wether there is a lack of information in the initial base (not enough counter-examples) or its conclusion really deserve such confidence.

The confidence of an initial experiment of a weakly coherent base is naturally 1. Eventual incoherent experiments that were forced to coherence will be reflected by the certainty factor. The confidence of 1 only means that no "guess" on unknown parts of the learning domain have been made.

IV.5 Certainty factor and probability of a conclusion.

We now want to take into account any incoherent or repeated experiment that "disappeared" when the initial base was forced to be weakly coherent; if the initial base is supposed to be representative of the learning domain, we will in fact quantify the probability of the conclusion of a rule (*a posteriori* probability, that is with known hypotheses).

The number of times an example appears in the initial base (even with contradictory conclusions) must influence the certainty factor.

This certainty factor is defined on the final extrapolated rules by decreasing order of precision.

We now refer by **A** to the very initial base, that is, before any conflict resolution in view of weak coherence.
We drop the j-subscript of the conclusion, knowing that a certainty factor is attached to a conclusion.

For any E, term of the initial base, let

N(E) be the total number of occurence of E in **A**, and
$N_R(E)$ be the number of times E appears in **A** with the same conclusion f than R, R being any extrapolated rule.

Définition 12 : *the certainty factor of conclusion f of a term* R *is defined by*

$$\text{Proba}(R) = \frac{\sum_{E \in A,\ f(E)=f(R)} Nb_R(E) \cdot \text{Proba}(E) \cdot \Phi_R(E \cap R)}{\sum_{E \in A} Nb(E) \cdot \text{Proba}(E) \cdot \Phi_R(E \cap R)}$$

If R is an isolated term of the initial base (not intersecting any other term), the certainty factor of any of its conclusion is, as wanted, the relative frequency of its occurence in **A**.

V FIRST RESULTS

The reduction and extrapolation algorithms have been implemented on PC-AT, in C (Instant-C, Rationnal Inc), and tested on artificial bases, randomly generated according to given boolean functions.

These validations lead to following remarks :

- The reduction needs less ressources (both in computing time and memory) than the extrapolation ;
- The extrapolation computing time mainly depends on the ratio rules/counter-examples, and on the base dispersion.
  The extrapolation can apply to initial learning base, although it is longer, and sometimes leads to more hazardous extrapolated rules.
- Hazardous rules are inevitably generated, when the learning base is not sufficiently large, or not representative of the whole learning domain ; the question is : are these rules simple enough to be easily refutable by an expert ?

V.1 100 experiments, 25 entries, 2 conclusions

Reduction : Time = 1'30"
Extrapolation : Time = 3'30"

The fixed rules are generated in 5'.
Two more rules appear.

- **Function 1** : 1'30" — **4 extrapolated rules**

If Not e_1 *AND* e_5, *OR* *If* e_6 *AND* e_{11}, *Then* f_1

The above rules are well recognized ; 2 more rules, cross-over terms, are supplied, which are avoided by addition of 2 counter-examples.

If Not e_1 *AND* e_6 *Then* f_1 ; *If* e_5 *AND* e_{11} *Then* f_1

- **Function 2** : 2' — **3 extrapolated rules**

If e_2 *AND* e_{13}, *OR* e_4 *AND Not* e_{17}, *OR* e_{21}, *Then* f_2.

These rules are recognized by the extrapolation.

V.2 200 Experiments, 30 Entries, 2 Conclusions

Reduction : Time = 6'
Extrapolation : Time = 14'

77% of the given rules are generated in 20'.
Two rules are altered.

- **Function 1** : 3' — **4 extrapolated rules**

If e_7 *AND* e_8 *AND* e_{14}, *OR* e_{12} *AND Not* e_{14}, *OR* e_8 *AND* e_{24}, *Then* f_1.

At first, the exact rule e_7 *AND* e_8 *AND* e_{14} becomes :

If e_7 *AND* e_{12} *AND* e_{14}, *Then* f_1 ; *If* e_8 *AND* e_{12} *AND* e_{14}, *Then* f_1

- *Function 2* : 11′ **6 extrapolated rules**

If e_3 *OR* e_4 *OR* (e_7 *AND* e_{12})) *AND* (*Not* e_7 *OR* e_{17}), *Then* f_2.

Similarly, rule e_7 *AND* e_{12} *AND* e_{17} is altered.

The initial rules are finally found out, by addition of 3 counter-examples.

V.3 500 Experiments, 50 Entries, 2 Conclusions

Reduction : Time = 27′.

The extrapolation is stopped because of memory requirement.

VI CONCLUSIONS.

We have presented here a complete process of generalization of a learning base into a (hopefully) more concise and "easy to use" rule base.

Our system can be used for the construction as well as for the refinement of knowledge base : from a set of examples or rules, it builds a minimal lattice of explainable rules, and is then able to extrapolate it if the initial base is incomplete (in most large scale real problems).

But almost as important as learning is knowing what we have learned ; the validation of knowledge appears in two steps of the learning process :

- if the learning base is incoherent, a certainty factor on the conclusions of generalized rules gives the probability of these conclusions,
- if the learning base is incomplete, the certitude and the confidence of any extrapolated rule tells what part of hazard this rule contains, and how much this rule is to be trusted.

The feedback with the human expert is easy : the addition of a few counter-examples to the learning base can avoid exaggerated extrapolations, and their determination is straightforward as the system can explain where extrapolated rules come from.

The future directions for this work include handling of numerical entries, making no distinction between entries and conclusions, extending the notion of coherence to temporal sequences, and, from the user point of view, being able to suggest discriminating experiments, such as the intersections of extrapolated examples and counter-examples.

We would like to thank Professor J. Zarka for initializing and maintaining the study.

BIBLIOGRAPHY

[Bhatnagar 86] R.K. BHATNAGAR, L.N. KANAL, *Handling uncertain information: A review of numeric and non-numeric methods.* In Uncertainty in Artificial Intelligence (L.N.Kanal, J.F.Lemmer ed.) North Holland 1986.

[Buchanan 84] B.G.BUCHANAN, E.H. SHORTLIFFE, *Rule based expert systems : the Mycin experiments on the Stanford heuristic programming project.* Addison Wesley , Reading, Mass., 1984.

[Diday 76] E. DIDAY, J.C. SIMON, *Clustering Analysis.* Communication & Cybernetics 10, K.S. Fu Ed, Springer Verlag, Heidelberg, New York, 1976.

[Ho Tu Bao 87] HO TU BAO, E. DIDAY, M. GETTLER-SUMMA, *Generating rules for expert system from observations.* 7th International Workshop, "Expert Systems and their applications", Avignon 1987.

[Hong 84] S.J. HONG, R.J. CAIN, D.J. OSTAPKO, *Mini : a heuristic approach for logic minimization.* IBM journal R&D, Vol 18, Sept 1974.

[Kodratoff 85] Y. KODRATOFF, J.G. GANASCIA, *Démonstration automatique de théorêmes et systèmes experts.* 5th International Workshop, "Expert Systems and their applications", Avignon 1985.

[Kodratoff 86] Y. KODRATOFF *Leçons d'apprentissage symbolique.* Capadues Ed, Toulouse 1986.

[Laurière 86] J.L.LAURIERE, *Intelligence Artificielle, résolution de problèmes par l'homme et la machine.* Eyrolles Ed, 1986.

[Lenat 76] D.B. LENAT, *AM : An artificial intelligence approach to discovery in mathematics as heuristic search.* Ph D. dissertation, Stanford University, Stanford, California, 1976.

[Michalski 84] R.S. MICHALSKI, R.E. STEPP *Learning from observations : Conceptual Classification.* In Machine Learning : An Artificial Intelligence Approach, (R.S. Michalsky, J.G. Carbonnel, T.M. Mitchell Eds), Palo Alto, 1984.

[Mitchell 82] T.M. MITCHELL, *Generalization as Search.* Artificial Intelligence N° 18, 1982.

[Quinlan 84] J.R.QUINLAN, *Learning efficient classification procedures and their application to chess end game.* In Machine Learning : An Artificial Intelligence Approach, (R.S. Michalsky, J.G. Carbonnel, T.M. Mitchell Eds), Palo Alto, 1984.

[Saucier 87] G. SAUCIER, M. CRASTES de PAULER, S. HANRIAT, *Asyl : A rule-based synthesis tool.* 7th International Workshop, "Expert Systems and their applications", Avignon 1987.

[Sebag 85] M. SEBAG, *Cube, optimisateur de P.L.A. : Manuel utilisateur.* Thomson Informatique Services, Janv 1985.

[Sebag 87] M. SEBAG, M. SCHOENAUER, *Réduction, extrapolation et génération de coefficients de vraisemblance à partir de règles ou d'exemples.* Rapport interne, C.M.A.P. - Ecole Polytechnique, Décembre 1987.

KNOWLEDGE ACQUISITION FOR A KNOWLEDGE BASED IMAGE ANALYSIS SYSTEM*

S. Schröder, H. Niemann, G. Sagerer
Lehrstuhl für Informatik 5 (Mustererkennung)
Martensstr. 3
D-8520 Erlangen
Federal Republic of Germany

Abstract:
For the analysis of complex images or image sequences explicit knowledge about the task domain is required in order to derive a symbolic description of the input image data automatically. In this paper, a general approach for such image analysis systems based on an semantic network structure is described. It combines a powerful knowledge representation formalism with a control algorithm, which may be refined for special tasks. So far it was successfully used for the analysis of scintigraphic image sequences of the heart. Applications for the analysis of three-dimensional industrial scenes and speech understanding are in progress. An extremely important problem is knowledge acquisition which often is done manually. The main point of this paper is the description of general tools for the automatic acquisition of knowledge in the network mentioned above. These tools are not restricted to a special field of problems. Presently we use automatic knowledge acquisition for the analysis of industrial scenes for a flexible assembly system.

1. Introduction

The purpose of image analysis is the automatic computation of a symbolic description of an image, or an image sequence. The content of the symbolic description must correspond to the requirements of a particular application and provide a description on an adequate level of abstraction. In industrial scenes, for example, a description of the quality of manufactured objects or special tasks for a robot is required. To achieve this, complex applications require the explicit representation of extensive task-specific knowledge about objects that have to be recognized, their relations to each other and the properties of their task domain, that is the model. Furthermore there is a need for control algorithms to use the stored knowledge during the analysis process. The control algorithms have to be able to deal with uncertain terms of knowledge like dark, light, big, and small, and the representation formalism has to allow such uncertainties. The efficiency of an image analysis system therefore depends on both the knowledge base and the control algorithm. Besides the two modules knowledge base and control, a database for the intermediate results as well as methods of preprocessing and segmentation of images are necessary /NIE 81a/.

The ability of automatic knowledge acquisition (learning) is very important for modern image analysis systems. Although the analysis system is designed modularly, the adaptation to another task domain is a time consuming process. The bottleneck of this process is the construction of the new knowledge base.

There are basically two approaches to knowledge acquisition: leave it to the designer, that is, the manual approach, or leave it to the machine, that is, the (automatic) learning approach. And there is an arbitrary number of

*The work reported in this paper is supported by SIEMENS AG, Erlangen

related approaches resulting from some kind of mixture of the manual and the learning approach, that is the interactive knowledge acquisition. In the manual approach a sufficiently experienced, intelligent, and patient person looks at the problem and tries to write down, in the chosen representation formalism, the required knowledge. It is safe to say that there would be no image analysis without this approach, but of course great efforts are made to replace it by the learning approach. Probably it is reasonable to assume that in the interactive approach the importance of automatic learning will increase /NIE 81a/. Examples of interactive knowledge acquisition for image analysis systems are given in /PER 86a, YAS 77a/.

In the learning approach of symbolic descriptions you can distinguish two types /NIE 87a/:

1. Learning of a single situation. In this case the model will contain exactly 1 element (e.g. a 3D-model of a workpiece)
2. Learning of a class of situations. In this case the model will have many elements (e.g. a model of a car learned from several different images of different cars)

The second type can be divided into supervised and unsupervised learning. In the case of supervised learning the correct class or the correct concept name is known whereas in the case of unsupervised learning the system itself has to find out the similarities and it has to group the model accordingly.

As it was already mentioned above, the result of an image analysis system has to be a description on an adequate high level of abstraction. Of course there are several levels of abstraction represented in the knowledge base and there is always 1 level which is closed to the input data. Learning the knowledge on this level directly from input data (examples) is a great facilitation. But on the other hand it is not efficient to learn the knowledge of a high level of abstraction from examples (e.g. learning an action plan of a robot from image sequences of real actions). Such a knowledge should be transferred into the knowledge base using an explicit description (e.g. an action plan created by off line programming). There may be the problem that the high level knowledge is not explicitly formulated but that it does exist in the mind of some experts. In such a case at first this knowledge has to be extracted. An example of an intelligent tool for this process is given in /LIN 88a/.

In this paper a general knowledge acquisition algorithm for supervised learning of a class of situations or for learning of single situations will be described. This learning will be done by examples, that means segmentation results of input patterns, or by descriptions. It uses a semantic network for knowledge representation. The goal of our knowledge acquisition system is an almost automatic construction of some of the declarative part of the knowledge base for an image analysis system.

2. Knowledge Representation by a Semantic Network

Semantic networks are special, directed graphs, with labeled nodes and links. The basic idea is quite simple: Information about conceptions (objects, relations, events, situations, facts) are represented by nodes, interrelationships between nodes are represented by labeled links to also mark the kind of the relationship. The theoretical background and special implementations are given in /BRA 79a, LEV 79a/. In our opinion, such well structured networks with their capability to handle even procedural knowledge are an adequate way to build up knowledge bases for image (and speech) analysis systems. A complete knowledge based system can be integrated into such a structure /NIE 85a/.

The idea is that the semantic network with its nodes and links is a kind of skeleton specifying and representing the overall structure of the declarative knowledge base. Attached to each node there is local task-specific procedural knowledge. The declarative and procedural knowledge is activated by a problem independent control algorithm. Our definition of a semantic

network language described in this section is mainly influenced by KL-ONE /BRA 79a/ and PSN /LEV 79a/, but it has the following extensions making it particularly suited for pattern analysis and understanding:

1. The introduction of the **concrete** link in order to represent different levels of abstraction (or different conceptual systems).
2. The statement of precise rules for creating instances, thereby introducing a precise pragmatic of the network.
3. The formulation of problem-independent control algorithms.
4. The definition of sets of modality to distinguish obligatory and optional parts and/or concretes.
5. The possibility to represent context dependencies

By the network language and the associated control algorithms a complete system shell for knowledge based pattern analysis is given. This shell ERNEST is realized in the programming language C in an UNIX environment.

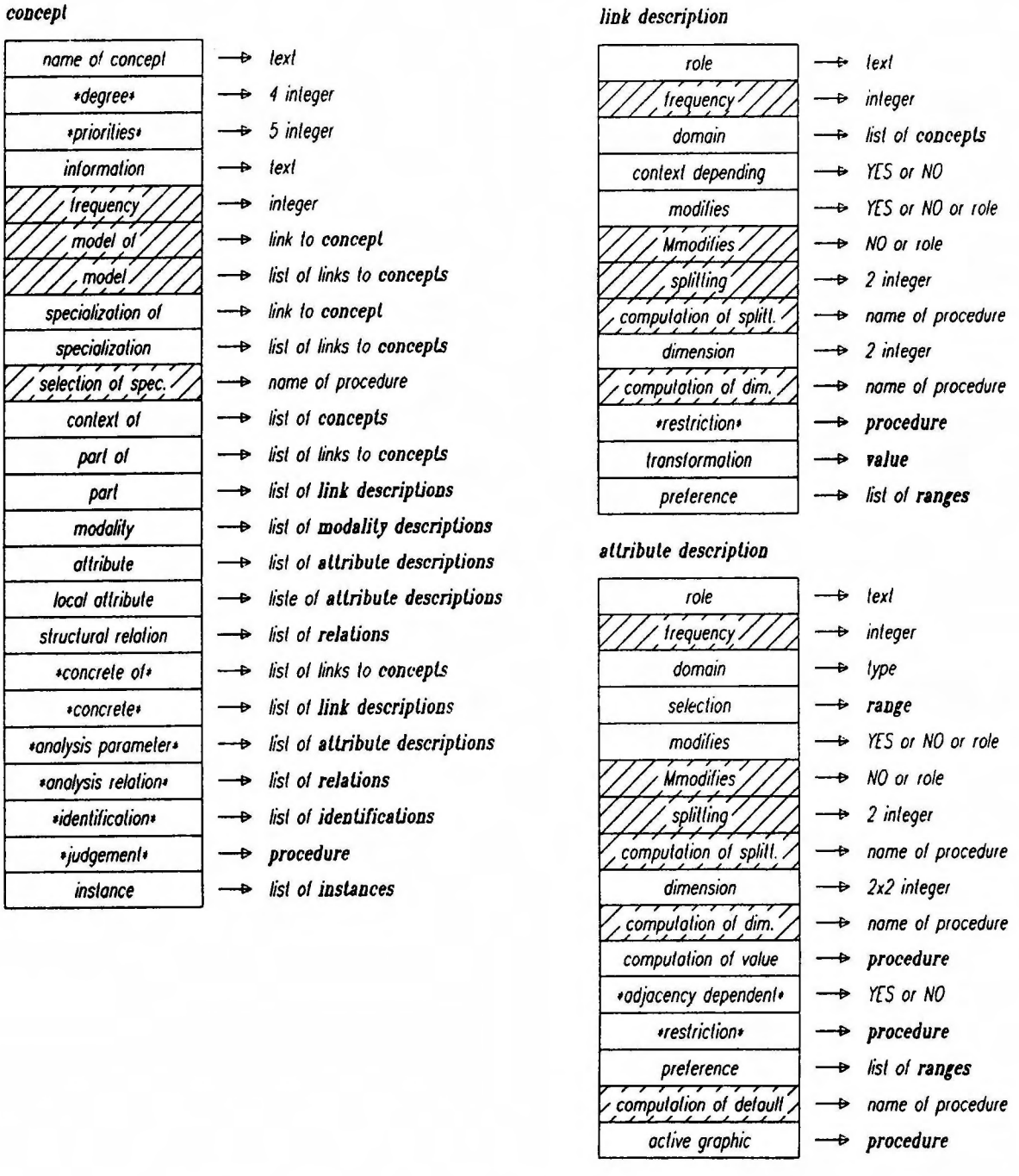


Fig.1: Concept Node, Link Description, and Attribute Description

Three different types of nodes exist in the network. First of all, a **concept** gives the intensional description of an object, an event, or classes of objects or events. The intensional description subsumes all properties of the conception to be modeled. Examples of such concepts are general ones like "3D-object", "movement", or "disease", or more special ones like "car", "run", or "enlarged left ventricle". The second type, called **instance**, represents a hypotheses that a real world object or event satisfies the definition of a concept. The concepts form the knowledge base, while their extensional sets, the instances, are the intermediate and final results. The third type, called **modified-concept**, is a concept that is restricted because of existing instances. Any details about the last two types are omitted here for brevity.

Relationships between concepts (and also between instances) can be built up by three different types of links. They are denoted by **specialization**, **part**, and **concrete**. Each of these links is described in a concept. The links define a hierarchy in the network. Both the part and concrete relationships are described by a substructure named **link description** that is shown in Fig.1. For all three relationships the inverse is also defined by **specialization-of**, **part-of**, and **concrete-of**, respectively. Furthermore, a node may have attributes, structural relations, and a certainty factor evaluating its reliability. Additional details, which have to be omitted here due to space limitations, are available in /SAG 85a, KUM 87a/.

It should be mentioned briefly that it is possible in the network described above to design a control algorithm which is independent of the task-specific knowledge represented in the network; it is only dependent on the special structure described above. Details are given in /SAG 85a, EIC 86a, SAG 88a/.

3. Knowledge Acquisition or Learning

The main idea of our knowledge acquisition algorithm is, that learning is done gradually. The first step is observation. Any sensor data has to be transformed into an internal representation. A new sample description is constructed during the second step whenever a new observation is made. During the third step the model is constructed by combination of the gained information. The general outline of the acquisition process, the goals of each step, and the data flow is shown in Fig.2.

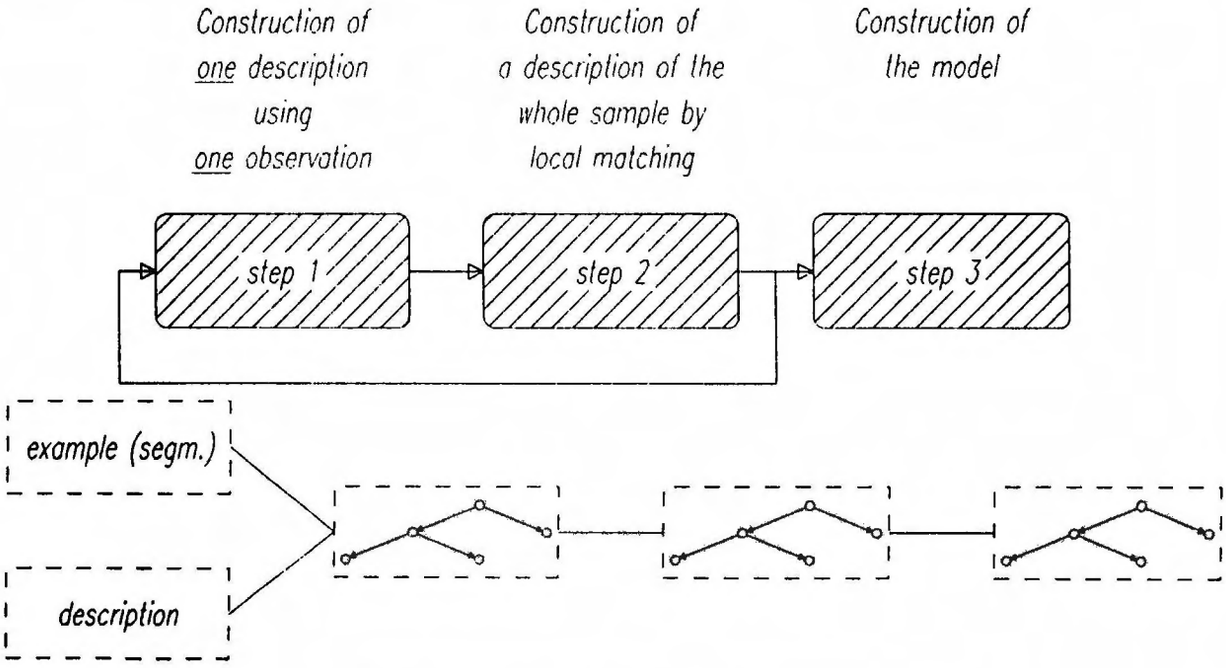


Fig.2: Outline of Acquisition Process

The system structure of the acquisition system is similar to the structure of the analysis system mentioned in the introduction. During the acquisition process the database for the results contains concepts instead of instances in the case of analysis. The control of the acquisition process is done by a special algorithm. Of course this algorithm is not as sophisticated as the control algorithm of the analysis process. The acquisition process does not have to classify the input data. The goal is to determine a symbolic description of the input data that can be used by the analysis process. The input data may be positive examples and/or descriptions. Learning the shape of industrial objects, the images of these objects are the positive examples whereas the CAD data form a description which can be used too.

The knowledge base of the acquisition process is based on the semantic network language. It is separated into a declarative and a procedural part. The declarative part is the model-scheme which is formed by concepts. It specifies the structure of the facts that you want to learn. In case of learning 3D-objects, the model-scheme defines the decomposition of such an object into surfaces, edges and vertices or volumes or whatever is possible to represent a 3D-object. The procedural part of the knowledge of the acquisition process consists of the procedures referred to by the slots **argument-test, computation of splitting, computation of dimension, computation of default,** and **selection of specialization** and of the rules used to compare intermediate results.

A concept is generated by the acquisition process using a special concept from the model-scheme called concept-scheme. This is indicated in the network by a link named **model** from this concept-scheme to the generated concept.

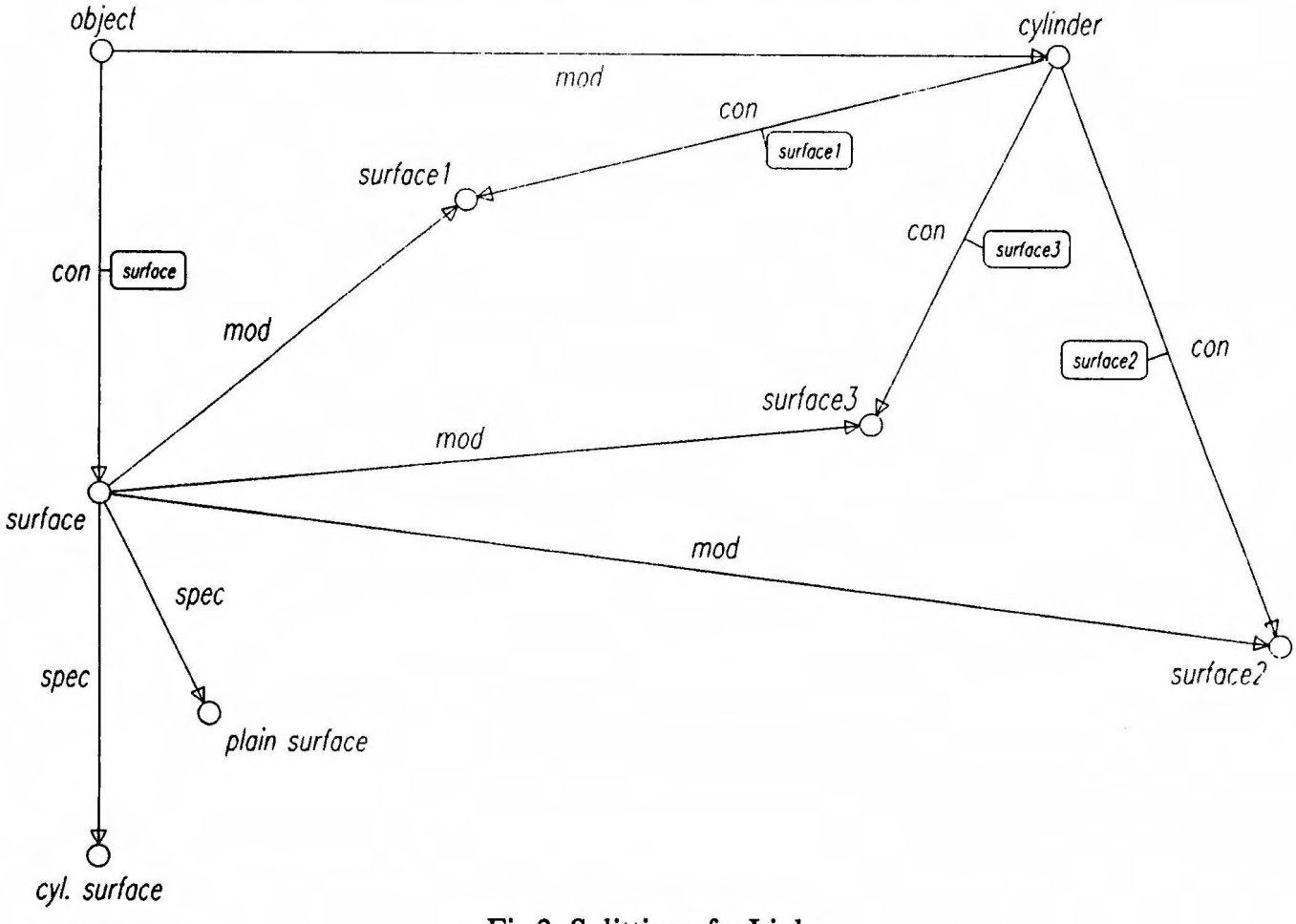


Fig.3: Splitting of a Link

During the first step of the acquisition process the existing parts, relations, and attributes of the concept-scheme may be split. For instance,

the concept-scheme 3D-OBJECT may have a part "surface". Of course, the generated concepts of real objects will have several surfaces. An example of splitting of a link is shown in Fig.3. To build up a model of a cylinder the concrete SURFACE has to be splitted three times. The procedure referred to by the slot **computation of splitting** computes the actual number of splittings. It is controlled by a given interval defining the allowed number. The same problem occurs in the case of dimension. Different model-concepts of the same concept-scheme may have different dimensions of their network structures. The problem is solved in the same way as it is done in case of the splitting. An item **mmodifies** is used if a network structure of the concept-scheme has to be split. This item then contains the role of the part, relation, or attribute of the concept-scheme. Otherwise **mmodifies** is set to NO.

If a concept-scheme has specializations, the appropriate one has to be selected. This is done by a procedure which is referred to by the slot **selection of specialization**. After having found the right specialization the link model-of/model will no more connect the further concept-scheme and the model-concept but the chosen specialization and the model-concept. This is shown in Fig.4. For the three existing surfaces of the model of a cylinder the concept PLAIN_SURFACE was selected two times and the concept CYLINDRICAL_SURFACE one time.

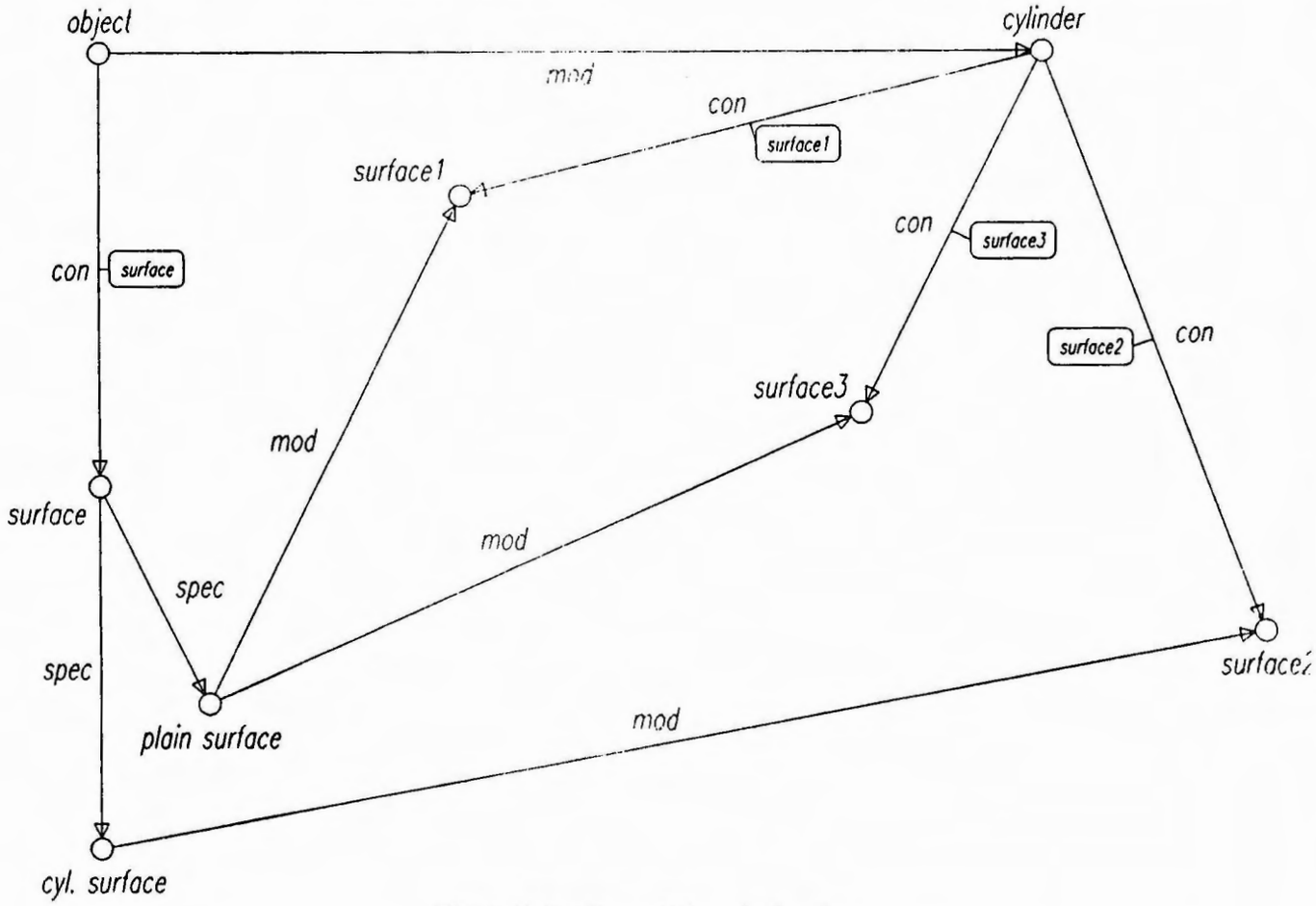


Fig.4: Selection of Specialization

During the first phase of step 1 the acquisition algorithm will build up a "concept-shell". This means that the new concept, its parts, attributes, and relations and the links between the parts and the concept are created, while the slots, which will later contain special values, are empty. During this phase computation of splitting and computation of dimension takes place.

The goal of the second phase is to test the existence of the arguments of the analysis procedures which are restricted by the syntax of the network

language. Only a concept which can be reached along one link or an attribute description of such a concept can be an argument for an analysis procedure. In the first case the argument will consist only of the role of the link, in the second case of the role of the link and the role of the attribute description. The test of the existence is done by calling the routines referred to by the slots **argument-test**. If network structures have been split, they will have new and different roles and they cannot have the role of the concerned network structure from the concept-scheme. In this case the names of the arguments which are composed of the roles have to be determined again. For example the judgement procedure of the concept-scheme OBJECT has only 1 argument, the role of the only concrete link. The judgement procedure of the generated concept of a cylinder (see Fig.3 and Fig.4) has three arguments, the roles of the three links that have been constructed by splitting. However it is required that the analysis procedures are able to work with different numbers of arguments but on the other hand, as a consequence of the acquisition process, the generated concepts can be used for analysis directly without adaption of the procedural knowledge. This is an important precondition for step 2 of the acquisition process.

During the third phase, the default value of each attribute is computed and the domain may be restricted with respect to some inaccuracy. Such a restriction can improve the analysis process, but if, for example, the analysis is supposed to recognize an object regardless of its position, it would be fatal to restrict the domain of position dependent attributes. Ranges can be restricted by definition of an interval, a region, a distribution or a fuzzy set. An example of learning fuzzy parameters can be found in the next section.

The concepts created by the first step are the intermediate results of the acquisition process that are stored in the database. A new sample description is constructed whenever a new observation has been transformed into the internal representation by the first step. The first phase of step 2 consists of a search for correspondences between the network structures. This matching is done using the analysis procedures referred to by slots of the concepts and of the substructures. The same procedures will later be used to analyze a whole scene with several objects. Indeed, matching in the case of knowledge acquisition is simpler than in the case of analysis. The new description has not to be compared with the whole knowledge base but only with the former sample description of the object that shall be learned. Of course the above described preconditions for the analysis procedures have to be taken into account.

During the second phase the frequency counters of the network structures, for which correspondences could be found, are updated. The other network structures of the new observation are added to the new sample description with a frequency counter value of 1. This second step is done to combine several observations of the subject of learning. Especially if you want to build up a complete 3D-model of an object you have to examine several images from different views. The fusion of the old sample description and a new observation is guided by the following rules:

RULE 1: fusion of 2 concepts C_1, C_2

IF there is a correspondence between C_1 and C_2
AND C_1 and C_2 have been generated using the same concept-scheme
AND the arguments of the judgement procedure can be fused or redetermined
THEN fuse C_1 and C_2, respectively, as follows:
- increase frequency counter of C_1
- try to activate rules 2-4 for all substructures with equal roles
- determine the best fusion for those substructures that have been splitted. Try to activate rules 2-4 respectively
- create a new modality description for C_1 with respect to the links of C_2
- copy all remaining substructures that can not be fused from C_2 into C_1

RULE 2: fusion of 2 link descriptions L_1 and L_2

IF there is a correspondence between the domains of L_1 and L_2
AND the concepts to which the link descriptions do belong have been fused
AND L_1 and L_2 have been generated using the same link description from the model-scheme
THEN fuse L_1 and L_2, respectively, as follows:
- increase frequency counter of L_1
- append the domain of L_2 to the domain of L_1
- append the preference of L_2 to the preference of L_1
- recompute the allowed interval for dimension
- for all those concepts from the domains for which there are correspondences, try to activate rule 1. If the concepts can be fused, redetermine the domain

RULE 3: fusion of 2 attribute descriptions A_1 and A_2

IF the concepts to which the attribute descriptions do belong have been fused
AND A_1 and A_2 have been generated using the same attribute description from the model-scheme
AND the arguments of the computation of value procedure can be fused or redetermined
THEN fuse A_1 and A_2, respectively, as follows:
- increase frequency counter of A_1
- append the preference of A_2 to the preference of A_1
- recompute the allowed interval for dimension
- recompute the restriction of the domain (selection) according to the selections of A_1 and A_2

RULE 4: fusion of 2 relations R_1 and R_2

IF the concepts to which the relations do belong have been fused
AND R_1 and R_2 have been generated using the same relation from the model-scheme
AND the arguments of the test of relation procedure can be fused or redetermined
THEN fuse R_1 and R_2, respectively, as follows:
- increase frequency counter of R_1

The third step will then create a new model by rule based comparison of the sample description. Such a comparison of concepts may lead to a description of a given class of objects (generalization). On the other hand, given only examples of single situation, segmentation faults may be corrected. Examples of the rules for step 3 are given in /DIE 81a/.

An optional fourth step may compare the created model with negative examples. The goal of this step is to verify and to specialize the model. It is

known from parameter learning in pattern analysis /NIE 81a/ that near misses will lead to the best results in discriminating patterns. But what are near misses in learning complex structural knowledge. What is a near miss of an airplane or a near miss of a ventilator. For a given application you may know the answer but in general it seems to be complicated to find these near misses. Therefore our knowledge acquisition algorithm will build up complete models with respect to the input data from positive examples and it is able to use near misses if there are some. The problem of near misses of real objects is discussed in /CON 87a/, too.

4. Example

A simple example of the described algorithm is the learning of an object model of a ventilator for a binary image system. The input data for the acquisition process is the result of the segmentation of a binary image of the ventilator. This binary image is shown in Fig.5. and the segmentation result is shown in Fig.6. The segmentation process computes features like contours, holes, vertices, centre of mass etc.

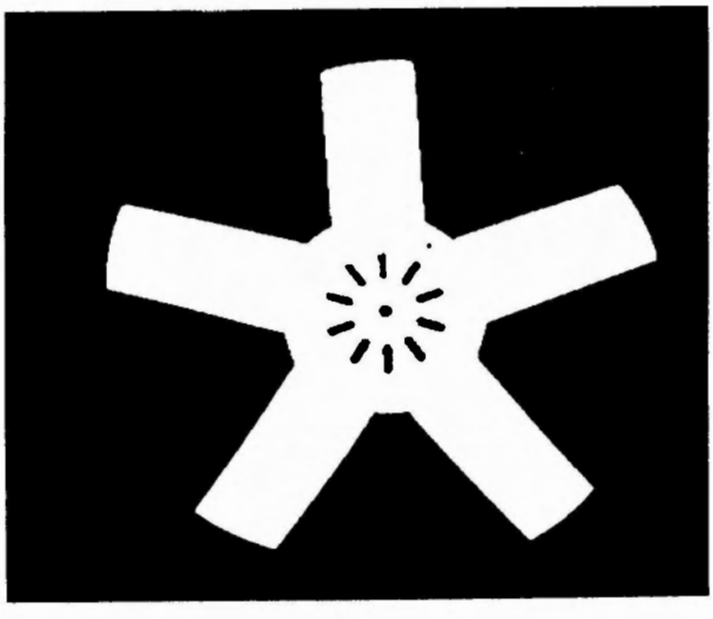


Fig.5: Binary Image of a Ventilator

The result of the first step of learning is a semantic network description of the input data. The model-scheme that is used by the acquisition process is shown in Fig.7. The scheme of an object is defined by one object surface and several or no holes. The object surface itself is defined by the surrounding contour that is built by line segments and by the vertices that section the contour into line segments. A line segment can be a straight line or a circle arc. Now the description of the image of the ventilator consists of 1 object surface (positive surface), 12 holes (negative surfaces), 11 line segments, and 11 vertices. Of course this description which is shown in Fig.8 contains the segmentation faults. These wrong concepts which are generated because of a wrong segmentation are underlined in Fig.8.

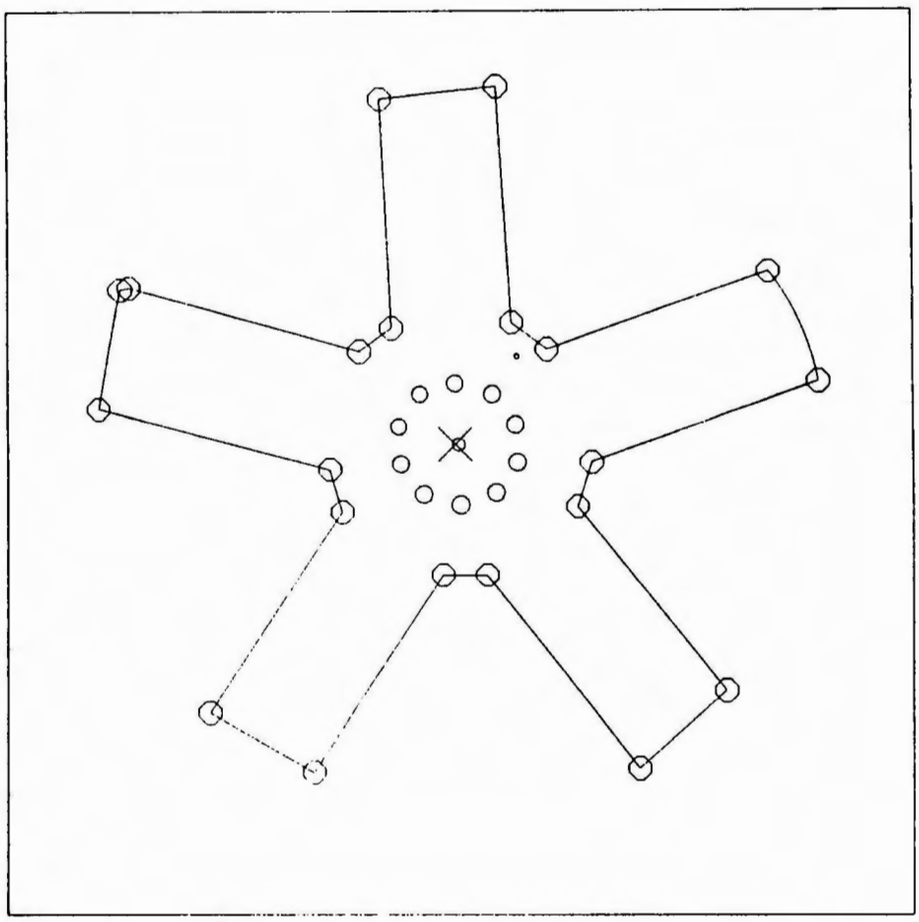


Fig.6: Result of Segmentation

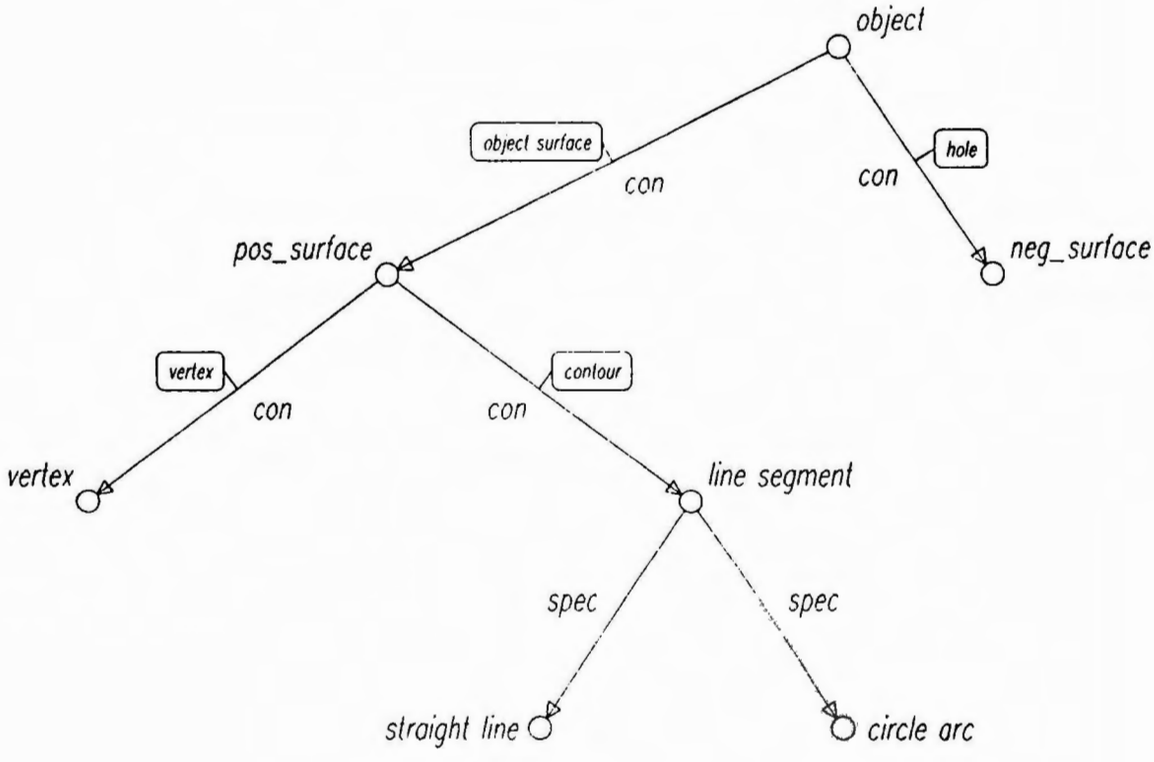


Fig.7: Model-Scheme

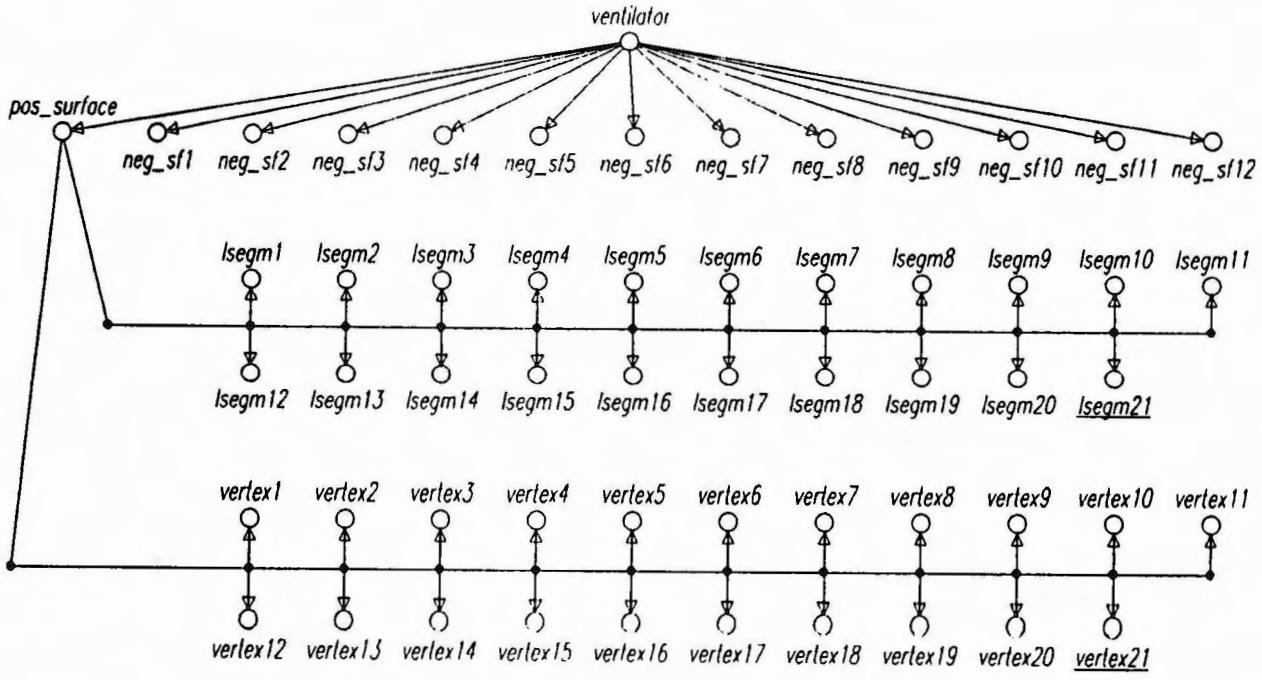


Fig.8: Result of Step 1: Network Description of Input Data

The first step will be repeated several times, each time with a different input image. The segmentation results of 10 different input images of the ventilator are shown in Fig.9.

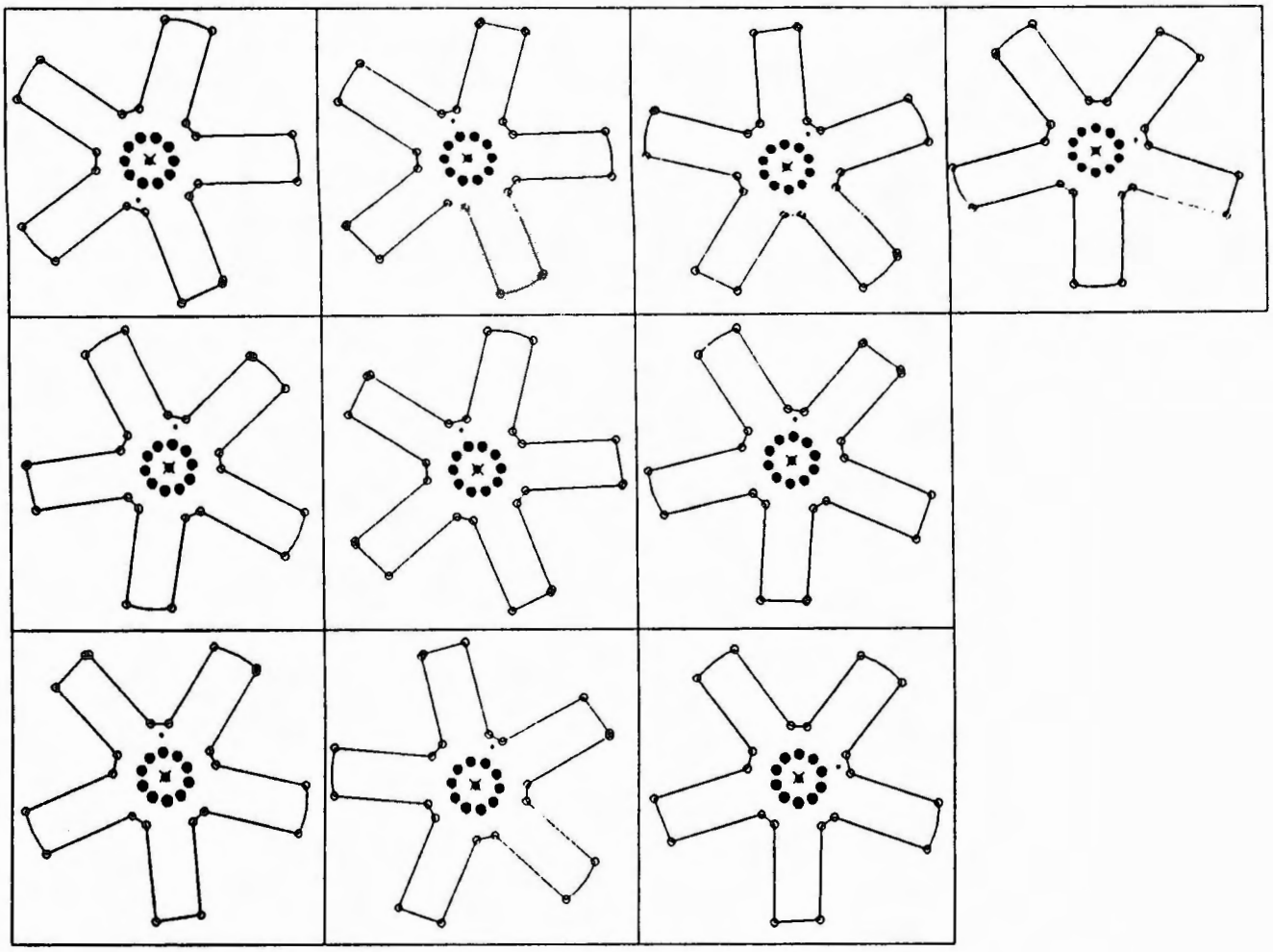


Fig.9: Segmentation Results of 10 different Images

Whenever step 1 is finished, the new description is combined with the former results of step 1. The constructed sample description looks like the description of Fig.8, but for every network structure the frequency will be counted.

The third step of the acquisition algorithm will then create the model of the ventilator by rule based comparison of the sample description. One of these rules requires that a network structure has to be purged, if its frequency counter remains under a certain threshold.

The final result - the model of the ventilator - is shown in Fig.10. Learning of 3D-object models or of models for grey scale image analysis will follow the same steps as mentioned above. Of course the segmentation process, the acquisition and analysis procedures will be more sophisticated.

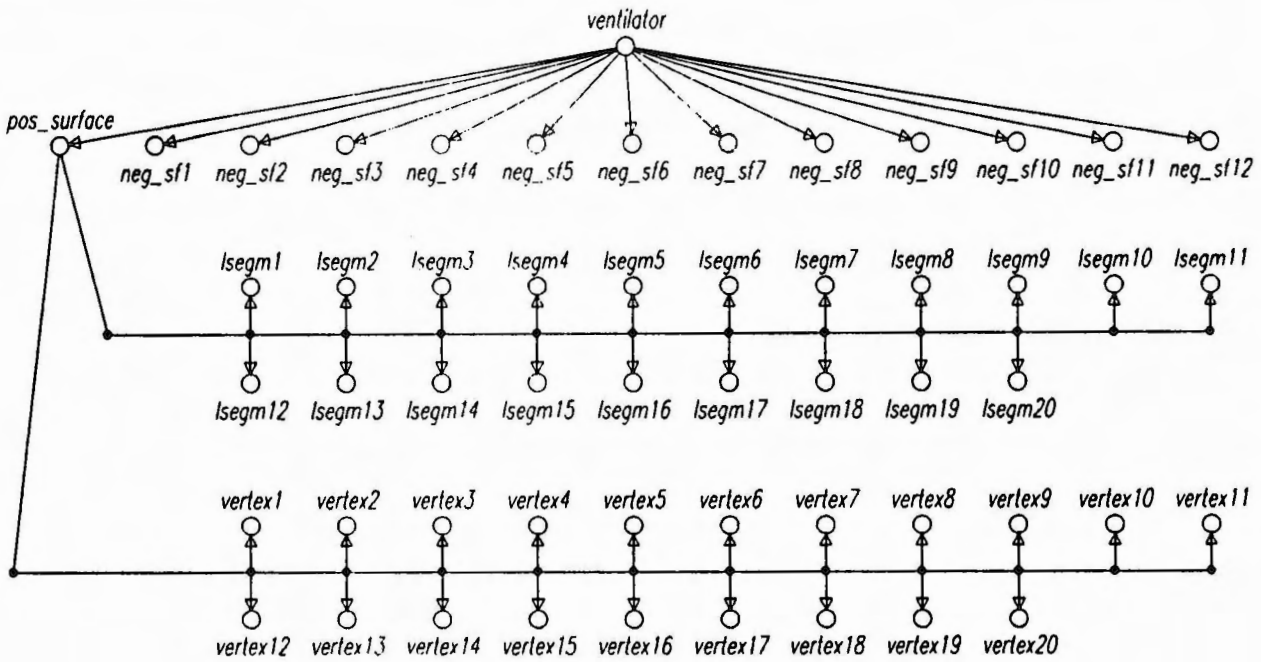


Fig.10: Model of Ventilator

The given example showed the way to learn network structures using our knowledge acquisition algorithm. But knowledge acquisition is not restricted to network structures. Parameters can be learned too. Important parameters for pattern analysis are those that restrict the domain of attributes (selection) because they will be used for the judgement of instances. It was already mentioned that it does not make sense to restrict every attribute and that uncertainties have to be taken under account.

We usually use fuzzy sets to represent such uncertain knowledge. The general idea of fuzzy sets is, that the range of the characteristic function u of a set A is not restricted to the values 0 and 1. Instead the range consists of the whole interval [0,1]. The function u is called fuzzy membership function. Similar to the standard set algebra operations for intersection, union, complement, and so on are defined for fuzzy sets /ZAD 75a/. There are algorithms to learn parameters for fuzzy membership functions based on observations /DEM 83a/. An often used kind of such a function has a trapezoid form. It is defined by four parameters.

The concept POS_SURFACE contains the attribute SHAPE FAKTOR sf. It is defined as follows:

$$sf = (S/L^2)*const$$

whereas S is the surface and L is the length of the surrounding contour. The

advantage of this definition which was taken from /YAS 77a/ is that the domain can be a priori restricted to values between 0 and 10. The resulting fuzzy parameters of the attribute in the model scheme are 0,0,10,10. Of course the fuzzy membership function defined in this way does not look like a trapezoid but like a rectangle. After every new observation the fuzzy parameters are changed to guarantee that a fuzzy value of 1 will result from each observed value. The observed value is appended to the list of preferences. During step 3 the fuzzy parameters may be recomputed using these preferences. The changes of the attribute description during the acquisition are shown in Fig.12 and the process of learning fuzzy parameters is shown in Fig.11.

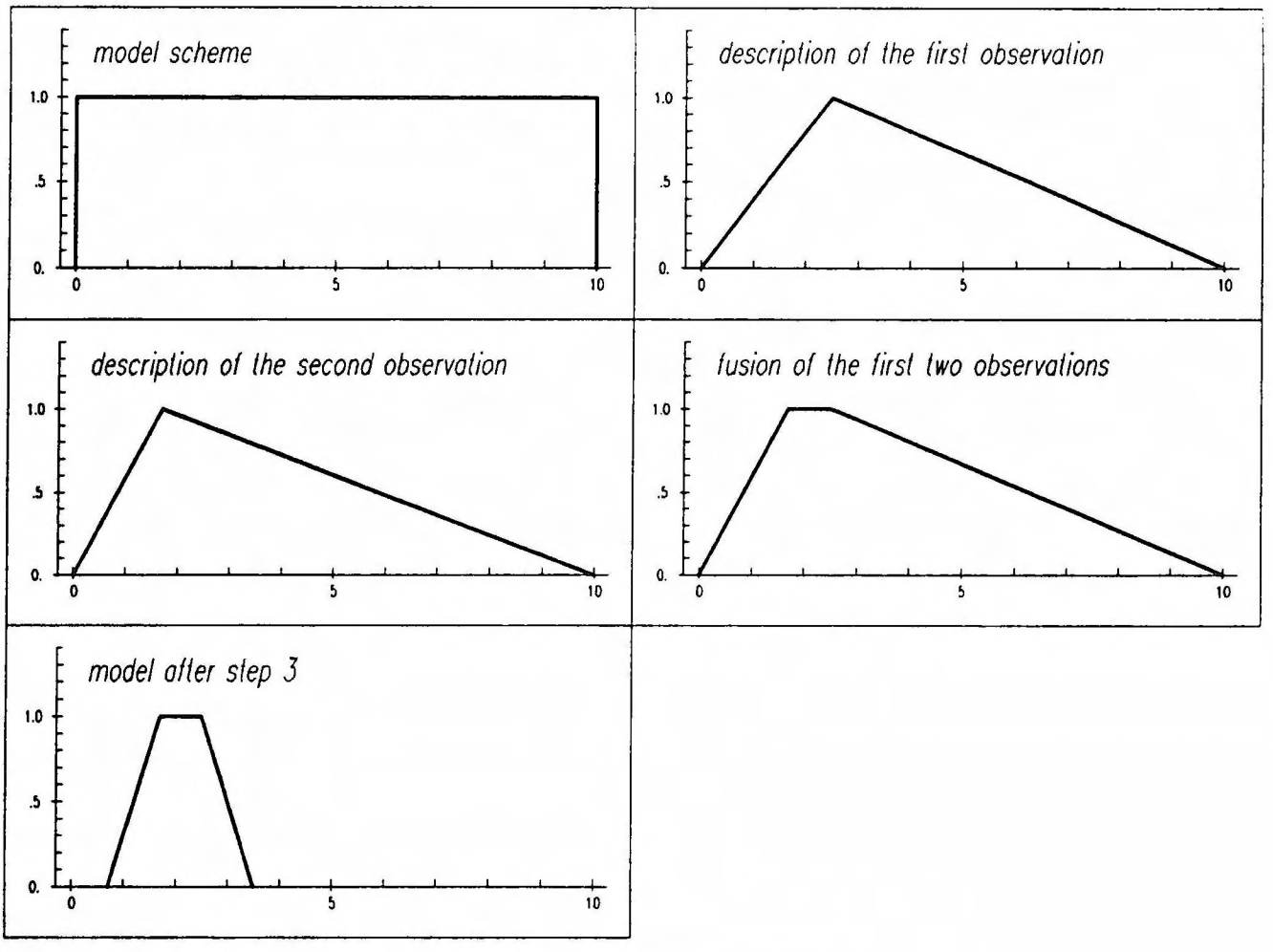


Fig.11: Learning of Fuzzy Parameters

5. Applications and Results

The semantic network language described was used successfully for the automatic diagnostic interpretation of scintigraphic image sequences of the heart. This system and its results are discussed in detail in /NIE 85a, SAG 85a/. A speech understanding system using the same approach is in development. The main point of the third application, interpretation of industrial scenes, is automatic knowledge acquisition. Step 1 and step 2 of the acquisition process have already been implemented and the implementation of step 3 is in progress.

The complete environment is realized in C under UNIX. The reason is the high efficiency. Other versions with specialized functionality were realized in FORTRAN /SAG 85a/ and LISP /EIC 86a/.

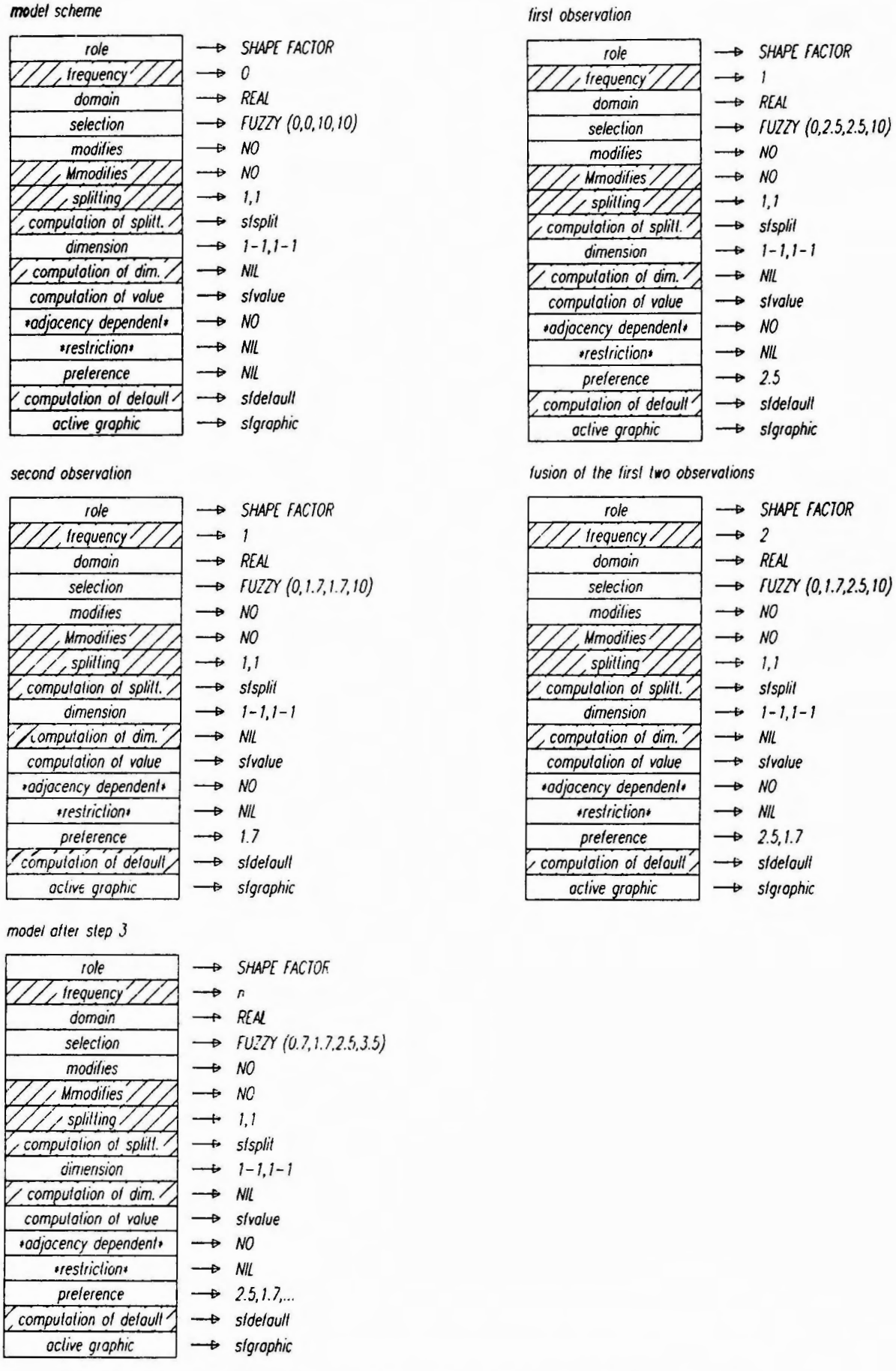


Fig.12: changes of an attribute description during acquisition

The acquisition algorithm presented in this paper was mainly influenced by the work of Connel and Brady /CON 85a, CON 87a/. They already have pointed out that former learning algorithms have not been able to learn visual shape descriptions from real visual data, because this data was unavailable or too complex and unstructured. Certainly this ability is important in the case

of learning for image analysis. But as it was mentioned in the introduction, only parts of the knowledge base can be learned from examples. Therefore our work extends the work of Connel and Brady in the way that the algorithm can be used for both, learning from examples, and learning from descriptions, and that it can use negative examples as well if there are any.

In addition the acquisition algorithm was strictly separated into a problem dependent and a problem independent part. The problem independent part- the control algorithm of step 1 to 4, the fusion rules of step 2 - has been integrated into a system shell for pattern analysis systems /KUM 87a/. This way we have got a general tool for knowledge acquisition for pattern analysis systems.

References:

/BRA 79a/ R.J. Brachman: On the Epistemological Status of Semantic Networks. In N.V. Findler (ed.): Associative Networks. Academic Press New York 1979,

/CON 85 a/ J.H. Connel, M. Brady: Learning Shape Descriptions. Proc. IJCAI85, 1985, 922-925

/CON 87a/ J.H. Connel, M. Brady: Generating and Generalizing Models of Visual Objects. AI, vol. 31, 1987, 159-183

/DEM 83a/ R. DeMori: Computer Models of Speech Using Fuzzy Algorithms. Plenum Press, New York, London, 1983

/DIE 81a/ T.Dietterich, R.Michalsky: Inductive Learning of Structural Descriptions. AI, vol. 16, 1981, 257-294

/EIC 86a/ W. Eichhorn, H. Niemann: A Bidirectional Control Strategy in a Hierarchical Knowledge Structure. Proc. 8. Int. Conf. on Pattern Recognition, Paris France 1986, 181-183

/KUM 87a/ F. Kummert, H. Niemann, G. Sagerer, S. Schröder: Werkzeuge zur modellgesteuerten Bildanalyse und Wissensakquisition - Das System ERNEST. In M. Paul (ed.): Proc. 17. GI-Jahrestagung, Springer-Verlag, 1987, 556-570

/LEV 79a/ H. Levesque, J. Mylopoulos: A Procedural Semantics for Semantic Networks. In N.V. Findler (ed.): Associative Networks. Academic Press New York 1979, 93-121

/LIN 88a/ M. Linster: KRITON, ein System zur Wissensakquisition für Expertensysteme. Der GMD-Spiegel 1/88, Bonn, 1988, 16-19

/NIE 81a/ H. Niemann: Pattern Analysis. Springer-Verlag, Berlin 1981

/NIE 85a/ H. Niemann: A Homogeneous Architecture for Knowledge Based Image Understanding. Proc. 2. Conf. Artificial Intelligence Applications. Miami Florida 1985, 88-93

/NIE 87a/ H. Niemann, H. Bunke: Künstliche Intelligenz in Bild- und Sprachanalyse. B.G. Teubner, Stuttgart 1987

/PER 86a/ W.A. Perkins: INSPECTOR: A Computer Vision System that Learns to Inspect Parts. IEEE Trans. Pattern Analysis and Machine Intelligence PAMI-5 (1986) 584-592

/SAG 85a/ G. Sagerer: Darstellung und Nutzung von Expertenwissen für ein Bildanalysesystem. Informatik Fachberichte, Springer-Verlag, 1985

/SAG 88a/ G. Sagerer, F. Kummert: Knowledge Based Systems for Speech Understanding. In: H. Niemann, M. Lang, G. Sagerer (eds): Recent Advances in Speech Understanding and Dialog Systems. Springer-Verlag, Berlin, to appear 1988

/YAS 77a/ M. Yashida, S. Tsuji: A Versatile Machine Vision System for Complex Industrial Parts. IEEE Trans. on Computers C-26 (1977) 882-894

/ZAD 75a/ L.A. Zadeh; Fuzzy Sets and Their Applications. In L.A. Zadeh, K.S. Fu, K. Tanaka, M. Shimura (eds.): Fuzzy Sets and Their Applications to Cognitive and Decision Processes. Academic Press, New York, 1975

INDE: A SYSTEM FOR KNOWLEDGE REFINEMENT AND MACHINE LEARNING

P.P. Terpstra and M.W. van Someren

Department of Social Science Informatics
University of Amsterdam

Herengracht 196
1016 BS Amsterdam
The Netherlands

Abstract
INDE is a system that generates classification rules from a (possibly incomplete) theory and refines the classification rules and the theory on the basis of counterexamples. INDE integrates Explanation Based Learning, Similarity Based Learning and Error Propagation.

1. Introduction

INDE is a system to be used during knowledge acquisition for refining and operationalisation of a knowledge base. A knowledge based system should provide efficient problem solving and be able to explain its solutions. These requirements can be in conflict. For economical problem solving the inference should make as few steps as possible, but an explanation should be detailed.
Also the development of a detailed model of the knowledge about a domain can be very elaborate, especially when one model is to be used for different tasks, i.e. repair and advice. INDE is an experiment to use machine learning techniques for refining the domain model.
The approach we take to resolve the mentioned complications (see also Morik, forthcoming and Steels et al. 1985) is to use two representations for the domain knowledge. (a) A descriptive domain theory, used for the explanation facility. (b) A set of shallow rules, which are operationalisations of the descriptive domain theory for efficient problemsolving in a particular task.
The shallow rules can be refined automatically in INDE by an inductive machine learning technique. The testresults and refinements of these rules are also used for searching for clauses in the underlying domain theory that are likely candidates for correction. The goal of this article is to show the integration of inductive and deductive machinelearn techniques and knowledge refinement in INDE.

2. Overview of INDE

Normally, the user first provides INDE (see fig. 1) a domain theory. This theory should be globally correct, but can be can be incomplete or false for some details. Next the user gives some training examples. INDE uses these examples with Explanation Based Learning (EBL) to derive shallow rules, which are reformulations of concepts, defined in the domain theory. These reformulations are expressed in terms, which can be applied with little or no computation.

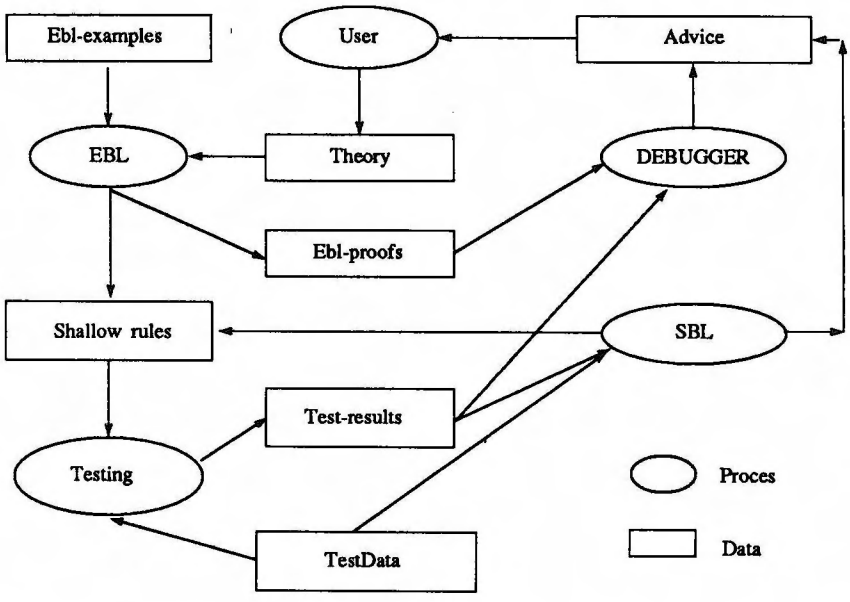


Figure 1: INDE

Next the user tests each shallow rule with a larger set of classified training examples. The testresults are used to refine the shallow rule by Similarity Based Learning (SBL).

The testresults of the shallow rules can also be used by the debugger to give the user advice which parts of the domain theory are candidates for modification.

3. Representation scheme

Observations, used as input for the problem solving, are represented as a collection of Prolog facts. The arguments can be atoms or numbers. Classification and theory rules are expressed by standard Prolog rules without cuts or other control statements. The rules can contain

- a Prolog term (with variables)
- a *range* (for a numerical value). The range is represented using the special predicates "<", ">", etc..
- a call to an arithmetic function
- a negated Prolog term.

The theory rules are, following LAS (Smith et al, 1986), annotated by "a justification type" and a declaration of the type of their arguments. With the justification types, the user expresses the confidence in each rule. They are used by the theory debugger. (see par. 5.3)
Problem solving by means of shallow rules is performed in one step, without a inference chain (except for negations in the conditions of a rule). All shallow rules are tried and all answers are collected. Inference in the domain theory is done in the standard Prolog way (depth first search and backtracking).

4. Cooperation between Explanation Based Learning and Similarity Based Learning

The Explanation Based Learning component was adapted from the algorithm by Mitchell et al. (1986). The SBL component is a version of the AQ algorithm (Michalski et al., 1986).
When a shallow rule is tested and appears to be overgeneral the rule has to be refined. SBL uses the testdata and testresults a examples to construct some extra conditions, a 'context' for the rule. The classification of the examples is defined by the testresults. Positive examples are those situations, where the rule gives a correct solution. Negative examples are those situations where the rules gives a wrong solution. The concept constructed by SBL is a generalization of those situations where the rule is correct. Because te EBL-rule serves as an initial hypothese, that directs the search of SBL for the final shallow rule, less examples are needed than in the case of SBL on its own.
When a shallow rule fails, but should be applied and it is not possible to use the domain theory to construct an alternative rule with EBL, SBL can construct the alternative rule.

4.1. Example of construction of an EBL-rule:

We tested INDE in a toy domain and provided INDE with artificiel examples. The task in this case is classification of weather situations by means of some local measurements to be used for shorttime weather forecasting.
The goal concept for EBL explains, why a cumulus cloud is not a raincloud. The domain theory states about this situation:
"A cloud with drops, which are smaller than 0.3 mm, does'nt cause rain. The size of a drop depends of the length of time that the drop stays in the cloud, when it flows with the convection current in the cloud to the top of the cloud. This calculation doesn't apply when the cloud is too wide. In that case the raindrops don't fall outside the cloud, but will flow back upwards with the convection current."

```
Theory:

1:empirical
  niceWeatherCloud(C):-
    convectionCloud(C),
    dropSize(C,S),
    S < 0.3.
2:definition
  convectionCloud(cumulus).
4:definition
   broad(W,H):-
     W > H.
```

```
3:empirical
   dropsize(Cloud,Size):-
     cloudWidth(Cloud,Width),
     cloudHeight(Cloud,Height),
     not broad(Width,Height),
     growSpeed(Cloud,Gs),
     convectionSpeed(Cloud,Cs),
     Size is Height*Gs/Cs.
5:assumption
     growSpeed(Cloud,0.000333).
```

The training example consists of some local measurements such as the kind of clouds, temperature, atmospheric pressure etc.

```
Trainingsexample:
6:observation
  cloudWidth(cumulus,1000).
8:observation
  convectionSpeed(cumulus,2).
```

```
7:observation
  cloudHeight(cumulus,1500).
```

EBL constructs with the domain theory the following proof that the observations are indeed a indication for nice weather:

```
Proof:
      Numbers denote the numbers of the used clauses.
      '<-' denotes  how a goal is proven.
      '='  denote the binding of an argument.

  niceWeatherCloud(C=cumulus)              <- 1 empirical
     convectionCloud(C=cumulus)            <- 2 definition
     dropSize(C=cumulus,S=0.24975)         <- 3 empirical
        cloudWidth(C=cumulus,W=1000)       <- 6 observation
        cloudHeight(C=cumulus,H=1500)      <- 7 observation
        not broad(W=1000,H=1500)           <- 4 definition
             W > H   /* this test fails */ <-   system
        growSpeed(C=cumulus,Gs=0.000333)   <- 5 assumption
        convectionSpeed(C=cumulus,Cs=2)    <- 8 observation
        S is H*Gs/Cs                       <-   system
     S < G. (G=0.3)                        <-   system
```

Out of this proof a shallow rule is constructed:

```
Classification rule:
  niceWeatherCloud(C):- convectionCloud(C),
                        cloudWidth(C,W),
                        cloudHeight(C,H),
                        not W > H,
                        convectionSpeed(C,Cs),
                        H * 0.000333/Cs < 0.3.
```

4.2. Refinement of the EBL-rule by SBL

The rule that was derived using EBL is in fact too general. From the testresults, SBL modifies this rule by adding the following context as an extra condition:

```
not ( atmosphericPressure(falling),
      windDirection(southWest));
( cloudEdges(clear),
  humidity < 70 ).
```

5. The theory debugger

During the derivation of the shallow rule EBL constructs a proof that the training example is an instantiation of the goal concept. If the shallow rule produces an error, one or more theory rules, used in the proof will be responsible. To indicate which theory rules are responsible, the debugger has to infer how errors in a theory clause are translated in errors in the shallow rule (error propagation).
So the theory is tested indirectly by testing the shallow rules. This has the advantage that testing of a shallow rule is more economically and that the use of shallow rules assures that every time (implicitly) the same part of the domain theory is tested. Also

SBL-contexts are a generalized description of the situations where the theory rules, used in the EBL-proof are right or wrong.

5.1. The errorpropagation theory

For the errorpropagation a meta theory is defined about the language in which the theory is expressed. This theory defines what sort of errors a possible for the theory rules and how these errors are propagated through a inference chain.
INDE currently knows the following types of errors:
(a) Clause errors:
- A clause is *too general* ('og') if it has too few or too weak conditions or too many disjunctions.
- Alternatively, a clause is *too specific* ('os') if it has too many or too strong conditions or too few disjunctions

(b) Value errors:
- Only for numerical values are error types defined. A numeric parameter can be too big ('oe') or too small ('ue'). For nominal parameters it is not possible to define domain independent abstractions of errors, besides simply 'right' or 'wrong'.

To be able to state how errors are translated in diverse inference steps the errorpropagation theory also defines what relations are possible between errors and what error relations belong to the results and the arguments of each operator in the language. We distinguish the relations 'same', 'opposite', 'brel' and 'srel' between errors.

```
    same(X,Y): X suspected of error type E <-> Y suspected of errortype E
opposite(X,Y): X suspected of error type E <-> Y suspected of opposite errortype OE
(The opposite of 'og' is 'os' and of 'oe' is 'ue')
```

Some examples of the use of these relations for some following operators in the language are:

```
Operator ':-'  :same(Head,SubGoals in Body)
Operator 'not' :opposite(Negation,SubGoal in Negation)
Operator '* '  :same(Result,Arguments)
Operator '/ '  :same(Result,FirstArgument)
                opposite(Result,SecondArgument)
```

The language in which the domain theory is expressed can contain tests if the value of some argument falls in some range. The ranges are expressed by '>' and '<' etc. If the test is suspected of an error the boundaries of the ranges should be made smaller or bigger. This is stated in the errorpropagation theory by the relations:

```
srel(Test,UpperBoundary) : Test suspected of 'og' <-> UpperBoundary suspected of 'oe'
                         : Test suspected of 'os' <-> UpperBoundary suspected of 'ue'
brel(Test,UnderBoundary) : Test suspected of 'og' <-> UnderBoundary suspected of 'oe'
                         : Test suspected of 'os' <-> UnderBoundary suspected of 'ue'
```

5.2. Computation of the errorpropagation

The errorpropagation theory denotes what the possible relations between errors in a proof are. These relations form a graph. The nodes of the graph are the formulas and terms. The edges of the graph are the relations. This graph has to be computed only once for each proof, because it is independent of the kind of error of the shallow rule.
Given the exact error of the shallow rule, this error is propagated through the graph, with as result a table that for each domain theory rule, used in the EBL-proof lists those possible errors which are consistent with the error of the shallow rule.

5.2.1. Example of the computation of the errorpropagation

The following graph is constructed out of the proof of the classification rule 'niceWeatherCloud' in par. 4.1:

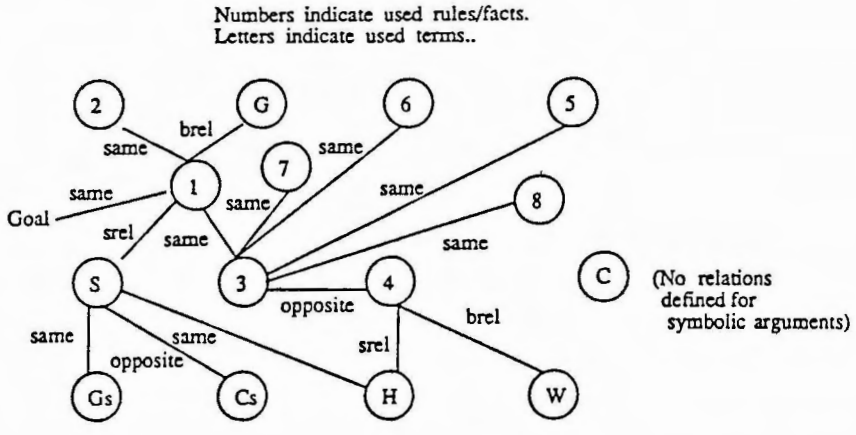


figure 2: Error relations in proof tree.

When 'niceWeatherCloud' rule succeeds, but it starts raining, the goal of its proof has the error type 'og'. By propagation of this error through the graph we get the following table:

nr	type	#used	#og	#os	arg1	#oe	#ue	arg2	#oe	#ue
1	empirical	1	1	0	C=cumulus	_	_	_	_	_
2	definition	1	1	0	C=cumulus	_	_	_	_	_
3	empirical	1	1	1	C=cumulus	_	_	S=0.24975	1	1
4	definition	1	1	1	W=1000	1	1	H=1500	1	1
5	assumption	1	1	1	C=cumulus	_	_	Gs=0.000333	1	1
6	observation	1	1	1	C=cumulus	_	_	W=1000	1	1
7	observation	1	1	1	C=cumulus	_	_	H=1500	1	1
8	observation	1	1	1	C=cumulus	_	_	Cs=2	1	1

The graph has a cycle, so rules and arguments are suspected of more than one error type. For example one can read in this table, that theory clause 2 is suspected one time to be overgeneral. Rule 3 is suspected to be overgeneral as well as overspecific and its the value of its argument 'S' is suspected to be underestimated as well as overestimated. The error propagation doesn't speak about nominal arguments, so there

is no computed suspection of 'C'.

5.3. Computing the most suspected domain theory clause

One EBL-proof is not enough to indicate the theory clauses, responsible for the errors of the shallow rules. So the results of the errorpropagation of the proofs of all shallow rules are combined to get a survey, that shows for each theory rule: (1) how many times the rule is used for the construction of a shallow rule and (2) how often the theory rule is suspected of each of its possible errors.

The final credit for a theory statement is also based on its justification type. The justification type is used to strenghten or weaken the suspections, computed by the errorpropagation. We distinguish the following justification types:
Definitions are assumed to be correct and therefore are not suspected of errors. *Observations* are not suspected of errors in the current version, but that can easily be changed, if necessary for a particular application. *Assumptions* and *empirical statements* are possible suspects. The current scheme suspects assumptions before empirical statements.
The heuristic, used to indicate the most suspected theory clause is:

```
emax(Rule)=The maximum number of times the rule is suspected of one of the
           errors, defined for the type of the rule.
used(Rule)=The number of times the rule is used for the construction of a
           shallow rule
score(Rules)=-(emax(Rule)-used(Rule)) * justificationtype factor
```

So the information that is presented to the user, is:
a. Which theory clause is most suspected.
b. What kind of errors the rules and their arguments are suspected of. The errortype also suggests how the rule can be improved. For example an argument, suspected to be too small, should be made bigger. Or a rule, that is suspected to be too general, should have an extra condition. In this case the SBL-contexts, constructed for the classification rules, that used the rule in their proof, can give a hint, what kind of condition that should be.

6. Related work

INDE is similar to LAS (Smith et al.), 1985). As far as we can tell, the main differences are that INDE

(a) is a fully implemented system
(b) acquires knowledge from a *set* of examples, rather than by discussing the implications of a single example with the user, and
(c) is able to acquire its classification rule even in the absence of an explanation using SBL.
(d) is meant to be used as an aid in constructing an expert system, where LAS claims to refine its theory during the normal use of the system.

INDE's debugger also resembles SEEK2 (Ginsberg 1986), but was developed separately. A difference is that SEEK2 tries to refine the theory automatically whereas INDE works interactively. Also SEEK2 does'nt use two representations for the domain knowledge.

7. Discussion

It will be clear from the description of INDE, that it is still in an early stage. We feel that the major contribution is to show the viability of several cooperating mechanisms for refinement.

INDE was developed using weather theory as an experimental domain. Experiments show that INDE produces sensible "suspect lists" in a number of cases, where errors were 'planted' in the knowledge base, but the experiments also show that the way of combining the results of errorpropagation in different proofs is too coarse. Eg. if a theory rule in proof A and also in proof B is suspected of being overgeneral, the kind of error of the rule in proof A can be different from that in proof B and it is not always correct to add these two suspections.
A better approach could be to relate an errortype with the context constructed by SBL for the shallow rule. Two suspections for a theory rule, computed out of two different EBL-rules should only be added if they have the same errortype a'd the corresponding contexts have some similarity.

A limitation of INDE's debugger is, that it performs much better for clauses that are too general than for clauses that are too specific and also it does not deal very well with negation. If a proof fails, the trace is much bigger, which makes the search problem much more difficult.

An interesting direction for further work is to extend the metalevel descriptions of the knowledge base. Currently INDE has only minimal metalevel knowledge, in particular about generality and simple arithmetic. For some domains it is possible (even natural) to specify richer metalevel knowledge about the knowledge base, which can be used to make the search for "suspects" much more powerful, eg. objects in the theory can be typed and part of the meaning of theory clauses can be expressed in these types.

References

Ginsberg A. (1986), Refinement of expert system knowledge bases: A metalinquistic framework for heuristic analysis. CBM-TR-147: Laboratory for Computer Research, Rutgers University, New Brunswick.

Michalski R.S, Mozetic I., Hong J., Larsson N. (1986), The AQ15 Inductive Learning System, An overview and Experiments, ISG 86-20: Dep. of Comp. Science, Un. of Illinois.

Mitchell T.M., Keller R., Kedar-Cabelli, S. (1986). Explanation based generalization: A unifying view. In **Machine Learning 1** , p47-80

Morik K. (forthcoming) Sloppy Modelling, Technical University Berlin.

Smith R.G., Mitchell T.M., Winston H.A., Buchanan B.C. Representation and use of explicit justifications for knowledge base refinement. In IJCAI-85

Steel L., Velde W. v.d. (1985). Learning in second generation expert systems, In Kowalik (eds) **Knowledge based problem Solving**, New Jersey 1985.

Filling in the knowledge acquisition gap: via KADS´models of expertise to ZDEST-2´s expert systems

Werner Karbach, Angi Voß [1]
F3-XPS, GMD
Schloß Birlinghoven

5205 St. Augustin

Xuejun Tong
AI-Laboratory
Computer Science and Engineering Dep.
Zhejiang University
Hangzhou, P.R.China

Abstract

Interpretation, that is discovering and describing the model of expertise, and its operationalization are two critical phases in knowledge acquisition. So far, there are approaches for each phase, but there are still isolated. We have combined two of them and think that it works fine because the two approaches are well developed and nicely fit together. The approaches in question are the KADS knowledge acquisition methodology and the ZDEST-2 expert system tool. The testbed for their integration was a mushroom classification problem.

Contents

1 Integrating two approaches to reduce the knowledge acquisition gap
2 A survey of the KADS knowledge acquisition methodology
3 Interpreting verbal data and building a model of expertise using the KADS methodology
4 A survey of the ZDEST-2 expert system tool
5 Operationalizing the KADS model of expertise into a ZDEST-2 expert system
6 Results

1 Integrating two approaches to reduce the knowledge acquisition gap

Knowledge acquisition is divided into three major phases:

1. knowledge elicitation
2. analysis and interpretation,
   where the structur of the problem solving process and the relevant types of knowledge are identified from the elicited data, in order to build an epistemological model of expertise.
3. operationalization of the interpreted and analysed data.

For the first phase elicitation techniques like interview. protocol analysis or repertory grid and scaling techniques have been developed (Breuker, Wielinga 84; Olson, Rueter 87).

The remainding phases are critical :
For phase 2 systems like ROGET (Benett 85) and OPAL (Musen et al. 86) were built, which incorporate a fixed, explicit model of the problem solving process and the application domain, respectively. Thus, they facilitate the interpretation of the elicited

[1] The work of these authors was partially funded by the Bundesministerium für Forschung und Technologie in the WEREX Verbundvorhaben under contract no. 413-5839-ITW8505DO

data which in term foccuses the progress of the knowledge elicitation process. As a drawback, such systems can only be used in restricted situations. Wanted is a more general support for the knowledge engineer building and describing arbitrary models of expertise.

When using tools like BABYLON, KEE or KNOWLEDGE CRAFT for operationalization, i.e. for developing a knowledge base, a great distance between the expert´s terminology and the knowledge representation formalisms must be covered. Therefore, higher knowledge representation formalisms are needed. (Gruber and Cohen 86) argue: "... that emphasizing the acquisition of knowledge in the design of an architecture is also good engineering; if knowledge acquisition tools are designed with the architecture, they can be integrated with runtime and implementation-level software. The function of knowledge acquisition interfaces is made easier when the underlying architecture supports task-level terms."

In order to support knowledge acquisition in its critical phases we have tried to integrate two approaches, one supporting the interpretation phase and the other the operationalization phase.

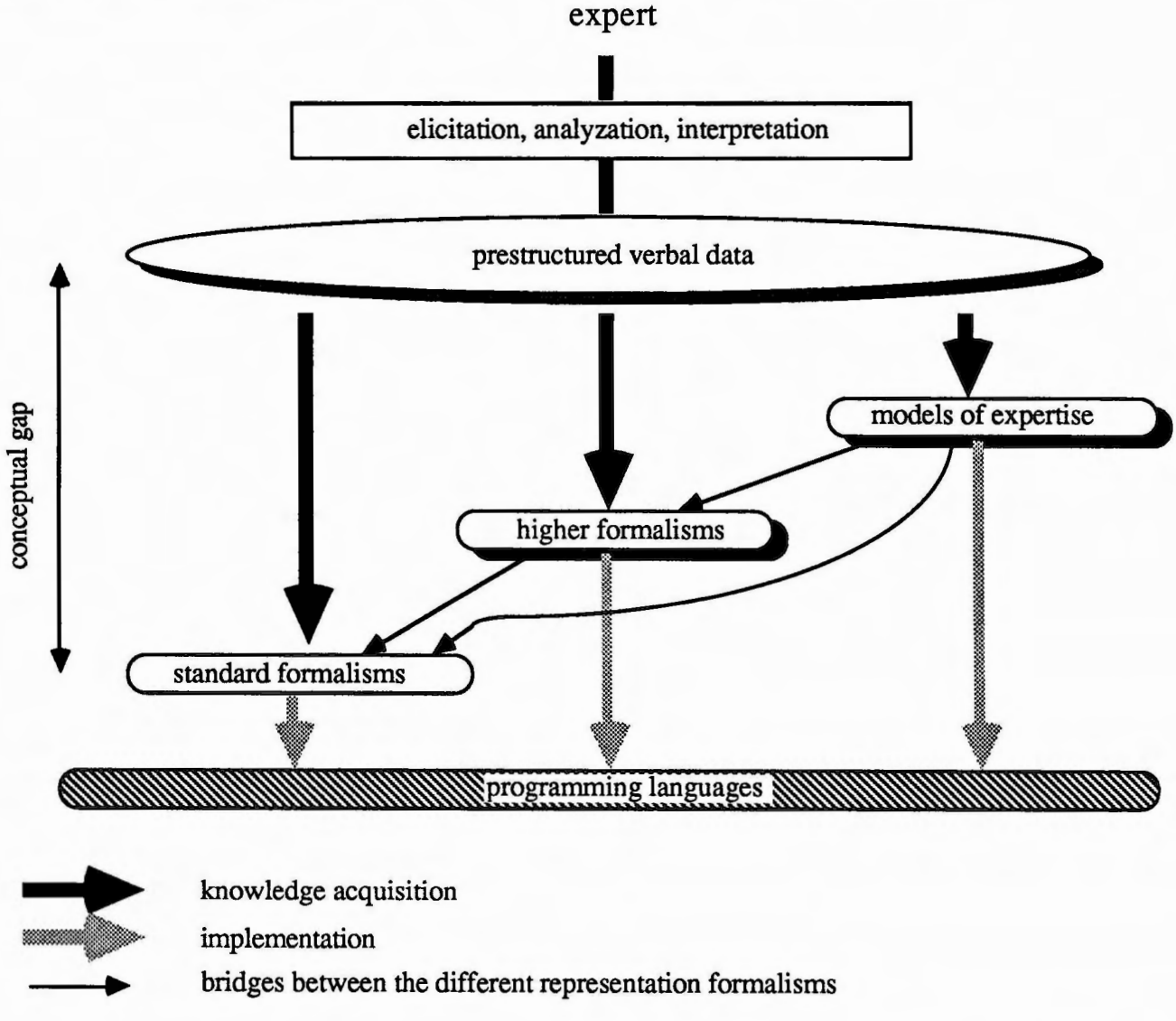


Fig. 1.1: the conceptual gap between expert and standard knowledge representation formalisms

The first is the KADS knowledge acquisition methodology which -in our view- is today´s most general elaborated approach for the analysis phase. It supports an explicit distinction and description of the problem solving process and the relevant types of knowledge on a epistemological level. There are suggestions for the operationalization phase, too, but they are still rather tentative.

Therefore we used the ZDEST-2 expert system tool (Tong, He, Yu 87) for operationalizing KADS´ conceptual model. ZDEST-2 knowledge representation formalisms are rather high level (as compared the tools mentioned before), and they are pretty close to KADS` conceptual models of expertise. Thus, as sketched in figure 1.1, KADS and ZDEST-2 try to narrow the conceptual gap from two opposite directions. By integrating these "good" approaches we think, the knowledge acquisition gap could be substantially reduced.

We tested these ideas with a mushroom classification application: Following the KADS methodology we obtained a conceptual model which we could transform into a ZDEST-2 expert system in a straightforward way.

2 A survey of the KADS knowledge acquisition methodology

The KADS group has provided a methodology for the developement of knowledge based systems which is based on three main principles (Breuker, Wielinga 85):

(1) the expertise should be analyzed before design and implementation starts,
(2) the knowledge analysis should be model-driven as early as possible, and
(3) the model of expertise should be expressed at the epistemological level.

As depicted in figure 2.1, the KADS methodology distinguishes four descriptions of expertise (Schreiber, Breuker, Bredeweg, Wielinga 88):

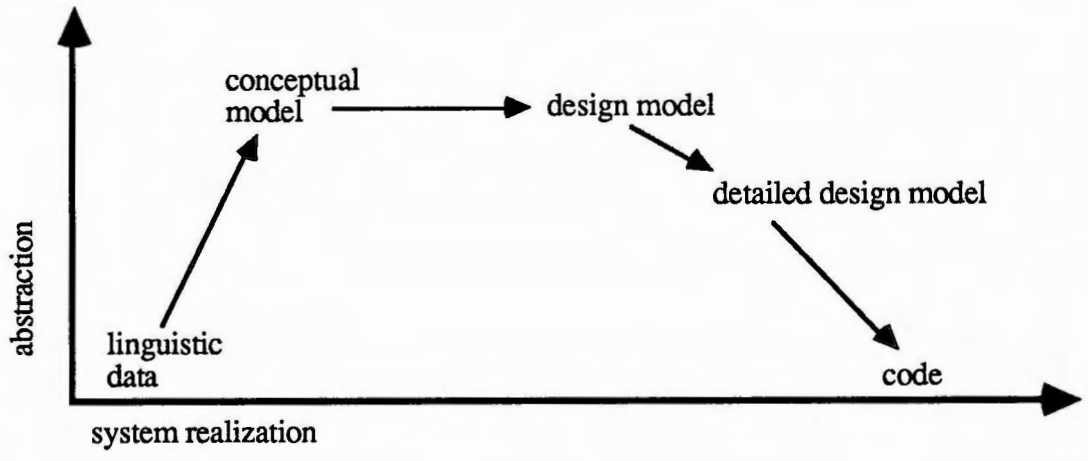


Fig. 2.1: KADS development of expertise descriptions

The four descriptions are divided into two phases, the **analysis phase** which has already been validtated in some experiments (A1 report 87), and the **design phase**, which is still very new, tentative and not fully worked out.

The *analysis phase* of knowledge acquisition starts with *linguistic data* as elicited from experts or literature. According to principle (3), they are analysed and interpreted to a four-layered model of the competence as observed in the domain expert. These *conceptual models* (Wielinga, Breuker 86) describe the expertise in the expert´s terminology so that he can discuss, criticize, and jugde them.

The conceptual model together with some *external requirements* about efficiency, costs, user interfaces etc. is input to the following *design phase*, where the knowledge based system is implemented via two further descriptions. The three-layered *design model* is the first model of the artefact and corresponds to what is called logical or abstract design in traditional design disciplines. It is refined to a *detailed* technical *design model* which can be transformed by simple mappings into the actual system. According to principle (1), the separation between conceptual model and design model is very strict: it should even be possible to develop them by different persons.

As far as we can judge from its current description, the design phase is far too liberal for our purposes, allowing unrestricted transformations of the conceptual model in the running system. Therefore we have adapted only the analysis phase from the KADS methodology. This brings us back to the conceptual models, which consist of four layers (please refer to section 3 for examples):

On the *domain layer*,
the static domain knowledge is described in terms of *concepts* and *relations*.

The *inference layer*
accomplishes an abstraction from the underlying domain layer: *Knowledge sources* describe possible inferences based on domain relations, and *meta-classes* abstract domain concepts to the roles they may play during problem solving. Knowledge sources and meta-classes form a directed graph, the *inference structure*.

On the *task layer*,
goal-directed *tasks* are specified which may be composed of subgoals and knowledge sources.

On the *strategic layer*,
rules are given to dynamically control, adapt or repair the execution of tasks.

It should be noted that in the KADS methodology, there is neither a formal language for the definition of the layers nor a fixed semantics. That means, the conceptual model is a paper and pencil specification whose understanding heavily relies on intuition.

As suggested in principle (2), the knowledge analysis process should be model-driven as early as possible. But what are the models guiding the analysis process?

The KADS group is developing a library of *interpretation models* which may be compared to the generic tasks in (Chandrasekaran 84). An interpretation model describes a generic, atomic expert task, such as diagnosis or monitoring. Being generic, interpretation models do not have any domain layer.

In real applications, the experts problem solving behavior is more complicated usually and combines several generic tasks. Therefore, interpretation models must be combined and modified to a so-called *real life model*. This, in turn has to be supplied with a domain layer in order to become a conceptual model.

Interpretation models and real life models specify the expert´s inference process with all domain-dependent peculiarities being abstracted away. They explicitly distinguish the different types of knowledge being needed to solve the problem. Therefore, they guide the knowledge engineer on categorizing the amorpous mass of elicited knowledge. Additionally, he recognizes missing knowledge (which is required in the model, but not yet elicited in his verbal data).
Thus, real life models and even more interpration models as their building blocks may play a central role in knowledge acquisition.

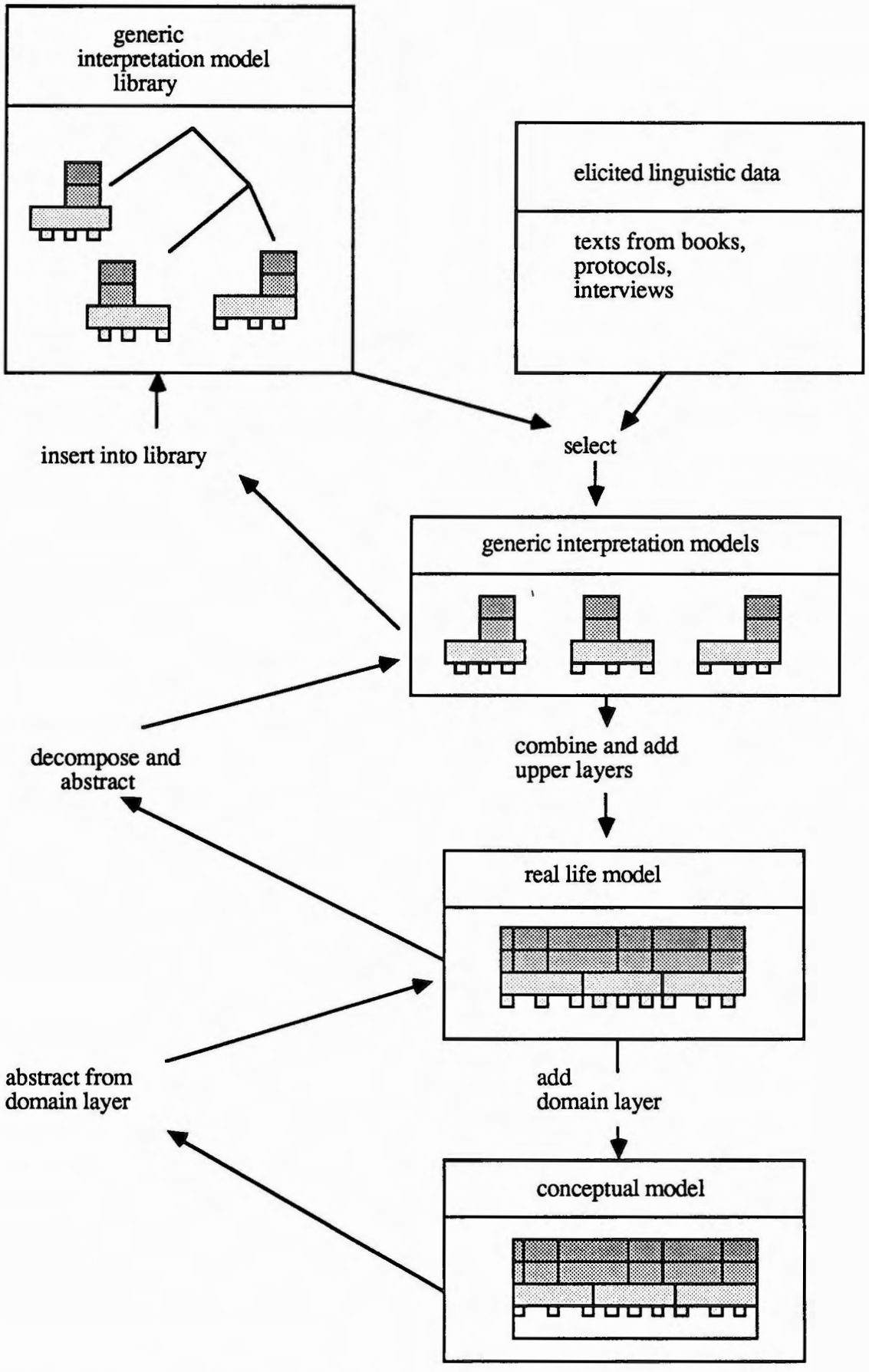


Fig. 2.2: KADS model life cycle

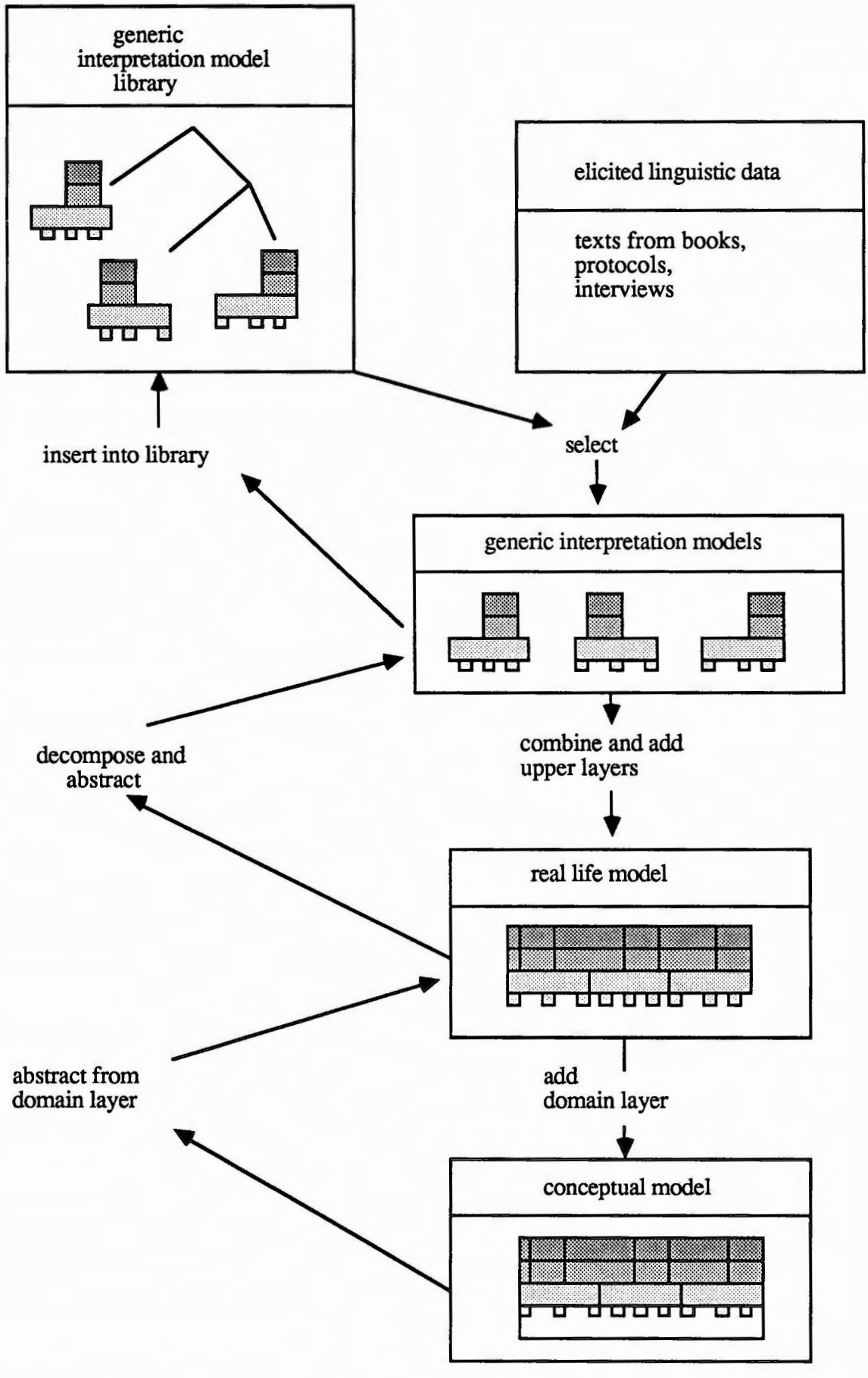


Fig. 2.2: KADS model life cycle

3. Interpreting the data and building the a model of expertise using the KADS methodology

Our source of expertise was a BABYLON knowledge base for mushroom classification (Müller 86). It embodies the following procedure:

> Some direct observations give rise to a set of first hypotheses, which is reduced to a single hypothesis by further, more expensive observations. For security reasons, this hypothesis is once more differentiated with confoundable mushrooms.

We developed the conceptual model along the KADS methodology as outlined in figure 2.2 above.

(1) Selection of an appropriate interpretation model:

As a starting point to characterize the problem solving behavior of the expert we chose the interpretation model for heuristic classification in the KADS library, which is based on the "horse shoe" model in (Clancey 1985). Figure 3.1 shows the inference level:

> Observables are abstracted to variables, which are heuristically matched to a solution abstraction or a set of hypotheses. This intermediate solution space is specified to get the solution(s).

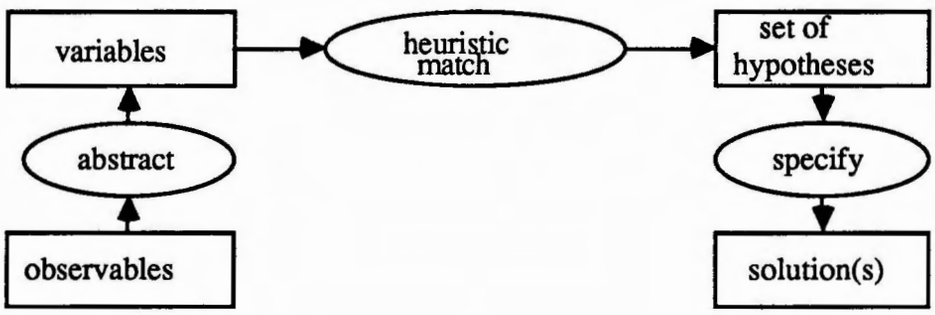


Fig. 3.1: inference structure of the KADS interpretation model for heuristic classification

This interpretation model does not exactly reflect the procedure the expert uses. To adapt it some modifications and extensions were needed.

(2) Modification and extension of the interpretation model:

Modifications:
In our procedure, the heuristic match is not made with variables, but with "direct observables", which in our domain are easy to detect without picking or damaging the mushroom.
As another difference, the specification of the single hypothesis is not based on variables, but on more "expensive observables", which are complementary to the direct observables.

Extensions:

In the KADS interpretation model for heuristic classification, a differential diagnosis confirming the solution is missing. So we had to add it:

From the metaclass "single hypothesis" we add a knowledge source to generate the confoundable mushrooms as "differential hypotheses". They are linked to a knowledge source establishing a new metaclass called "differential observables", which are differentiated to get the final solution.

Figure 3.2 shows the resulting inference structure of our model:

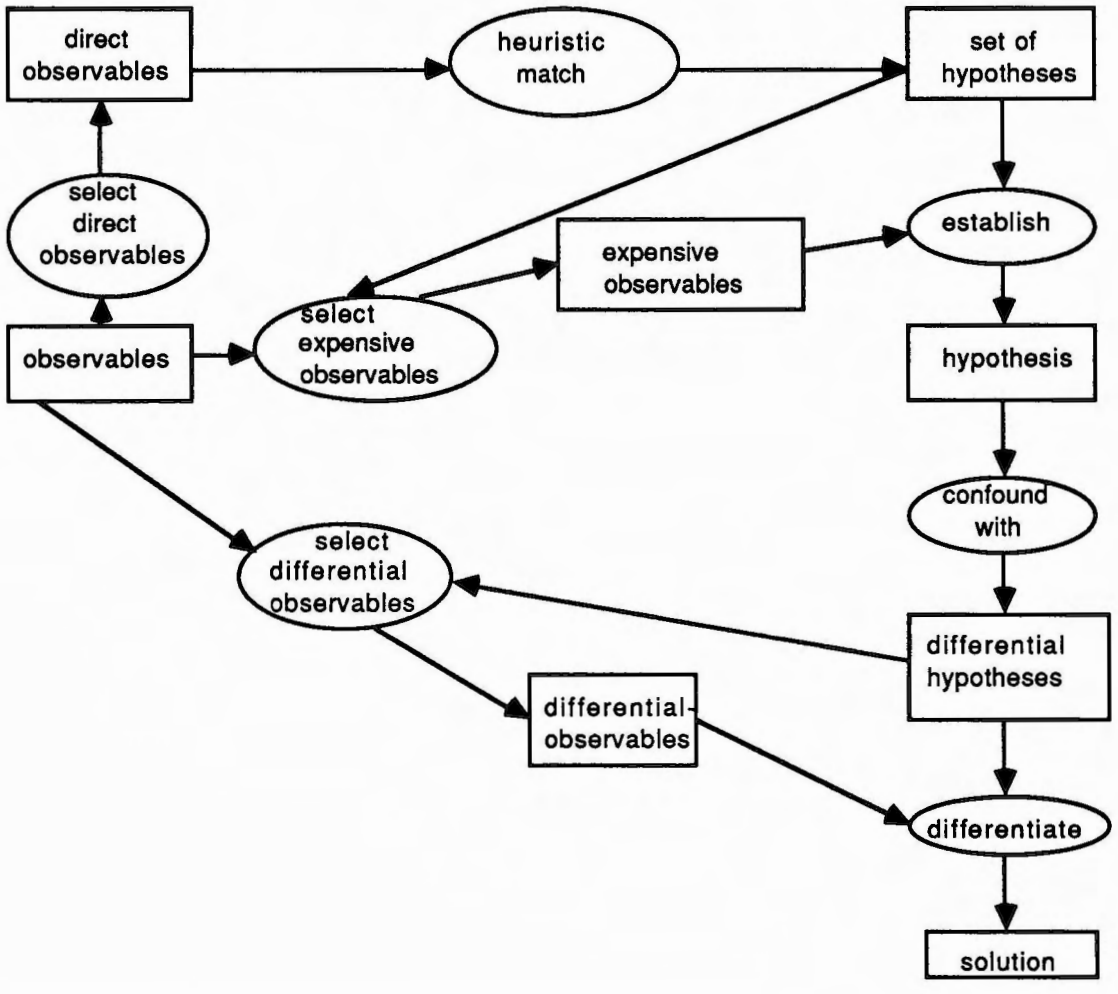


Fig. 3.2: inference structure for mushroom classification

(3) Attachment of task and strategic layers:

We organized the flow of control through our inference structure in a hierarchy of four tasks as shown in figure 3.3:

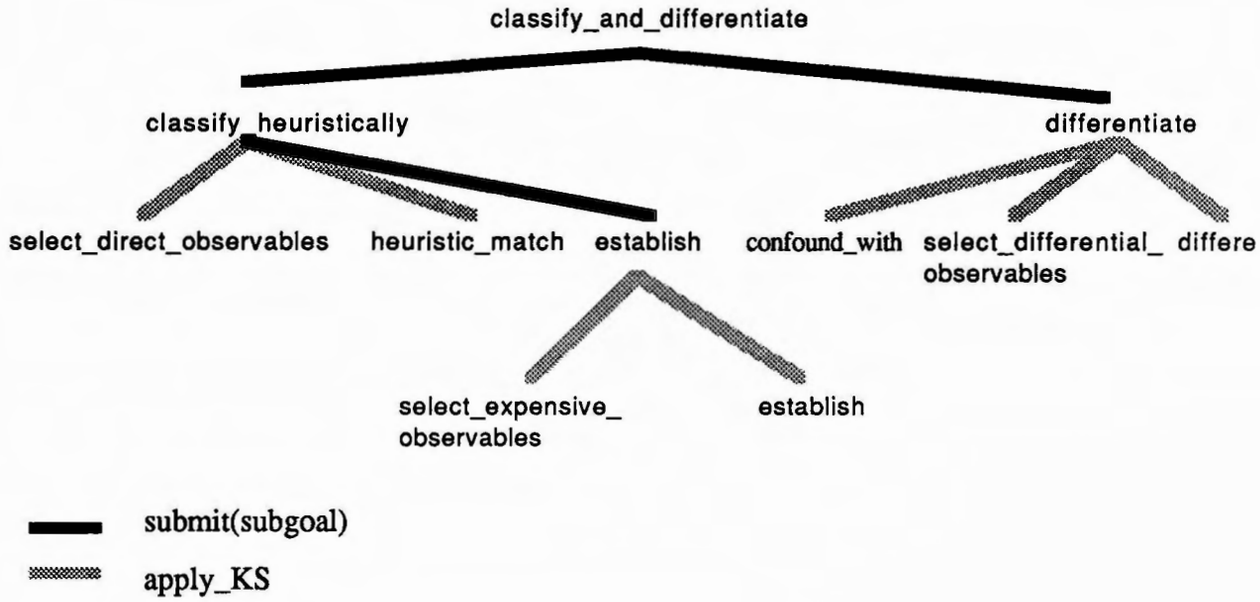


Fig. 3.3: call hierachy of tasks and knowledge sources.

Tasks:

classify_and_differentiate(object)
goal: classify(object)
method: heuristic
body: submit(classify_heuristically(set of hypothesis))
submit(differentiate(differentialhypotheses)

classify_heuristically(set_of_hypotheses)
goal: generate(set_of_hypotheses)
method: heuristic classification
body: apply_KS(select_direct_observables)
apply_KS(heuristic_match)
submit(establish(hypothesis))

establish(hypothesis)
goal: generate (hypothesis)
method: heuristic refinement
body: apply_KS(select_expensive_observables)
apply_KS(establish)

differentiate(differentialhypotheses)
goal: find(solution)
method: differential_diagnosis
body: apply_KS(select_differential_observables)
apply_KS(differentiate)

Since the tasks almost completely determine the flow of control, the strategic level is rather trivial. It consists of an agenda organized by the last-in-first-out (LIFO) principle.

Strategic, task, and inference layers together constitute the real life model of our mushroom classification problem.

(4) Attachment of domain layer:

The domain layer consists of concepts and relations. To attach it to the inference layer, we had to fill in the metaclasses by concepts and to describe the knowledge sources in terms of the relations. Thus, the inference layer gave us good cues about what knowledge should reside on the domain layer. Here is the interface between the two layers:

metaclasses:	**domain concepts:**
observables	all observables
direct observables	subset of observables
expensive observables	subset of observables
hypothesis	mushrooms
differential hypotheses	sets of mushrooms
solution	mushrooms

Knowledge sources:

select_direct_observables
"From the set of all observables, some directly observable ones are selected. In our case, this set was known beforehand, but it could also be generated dynamically depending on the application or the actual state in the problem-solving process."
Referenced domain relations: directly_observable

heuristic_match
"The direct observables are matched against a classification hierarchy to get a set of hypotheses."
Referenced domain relations: has_attribute

establish
"From a set of hypotheses a single hypothesis is established by "more expensive" observables."
Referenced domain relations: has_attribute

select_expensive_observables
see: select direct observables
Referenced domain relations: expensive_observable

confound_with
"A set of confoundable mushrooms is generated. This set is given for each mushroom on the domain level. "
Referenced domain relations: confound_with

select_differential_observables
see: select direct observables
Referenced domain relations: differential_observables

differentiate
"From the set of differential hypotheses a solution is determined by differentiating the set with differential observables."
Referenced domain relations: has_attribute

It will be sufficient to give just a few examples from the domain level:

Concepts:

mushrooms, boletus, taste, bitter, pleasant , colour_of_hat, brown red, ...

Relations:

has_attribute(boletus,form_of_pileus,.....)
direct_observable(boletus,form_of_pileus)
expensive_observable(boletus,taste)
confoundable_with(boletus,....)
...

Having attached the domain layer to the real life model, our conceptual model for mushroom classification is complete.

4 A survey of the ZDEST-2 expert system tool

The ZDEST-2 expert system tool was developed at the University of Zhejiang. It is currently running in Franzlisp on VAX and Honeywell machines, and its expert systems can be transferred to IBM PCs. So far, ZDEST-2 has been used to build an EMYCIN-like system called ZCLOT for detecting blood disorders, a KAS-like system called ZGPE for prospecting mineral resources, an expert system doing multi-factor planning to find the best composition of different kinds of cotton for cotton mills, and a trouble shooting system for radio repair.

A ZDEST-2 expert system or knowledge base consists of four levels:

The *conceptual level*
comprises static domain knowledge that is common to different experts. It is organized into a semantic network made of concepts and context types. *Concepts* describe basic domain objects. They are used as ingredients to define problems, rules, goals, hypotheses and messages on the hiher levels. Concepts are linked to form larger environments, the so-called *context types*, which, in turn, can be play the role of (super) concepts when their relation to other context types is described.

The conceptual level is split into a *surface level* for more heuristic knowledge and a *deep level* for more theoretical knowledge, although the distinction is admittedly rather pragmatical. For instance, in a diagnostic system for defects in cars, concepts like motor, engine, accelerator etc. might occur on the surface level, while the internal structures of motors, engines, accelerators etc. might be defined on the deep level. The deep knowledge is useful for explanations and to fall back to explicit reasoning in case where the surface knowledge is incomplete.

On the *concrete reasoning level*,
the conceptual knowledge is used to define *basic knowledge sources*. They describe how to solve primitive subproblems in terms of rules to be interpreted by a selected inference engine. There may be different basic knowledge sources for the same subproblem expressing how different experts would solve it.

On the *sketchy reasoning level*,
there are *meta knowledge sources* each devoted to a particular goal. They control and maybe modify the basic knowledge sources to adapt them to the current situation. Meta knowledge sources, too, consist of rules to be interpreted by a selected inference engine.

On the *planning level,*
tasks are defined by two rule sets, one for the precondition, and the other for the execution order of the meta knowledge sources they control. A task represents an elementary, self contained problem solving component, i.e. its input is a problem description and its output is a solution, which, however, may be used in other tasks. Thus, ZDEST-2 tasks correspond to the generic tasks in (Chandrasekaran 84).

While the sequence of basic knowledge sources within a meta knowledge source and the sequence of meta knowledge sources within a task is explicitly controled by rules, the execution of tasks is dynamically planned by a scheduler using a blackboard to keep track of the goals to be solved.

The knowledge on the conceptual level may be described in a semantic network representation language SNL. Rules too, may be described in SNL, but also in an EMYCIN-like expression language and in restricted natural language. The control components may not only be defined by rules with their inference engines, but also by procedures and functions in LISP, C, and PASCAL.

Due to its distinction between planning, sketchy and concrete reasoning levels, ZDEST-2 expert systems can be watched on three levels of control, which is exploited to give explanations on different levels of detail. Beside for explanations, ZDEST-2 provides facilities for knowledge acquisition, knowledge base refinement and assembly, and for maintenance.

5 Operationalizing the KADS model of expertise into a ZDEST-2 expert system

The levels and constructs in ZDEST-2 look much like the ones in KADS´ conceptual models. How easy would it be to transform the latter into the former? If we could find guidelines for this transition, we could front-end the KADS methodology for the anaylysis phase with the development of ZDEST-2 expert systems in the design phase, thus covering the entire knowledge acquisition process in a coherent way.

To check these ideas, we tried to convert our conceptual model for mushroom classification into a ZDEST-2 expert system. Following the steps sketched in figure 2.1, we first fixed the global architecture of a design description and then refined it to detailed design description which, in our case, coincided with the implementation. We resume the numbering from section 3:

(5) Global architecture

Since ZDEST-2 already provides a four level structure, we just tried to assign each notion in the conceptual model to one of these levels. This turned out to be surprisingly simple:

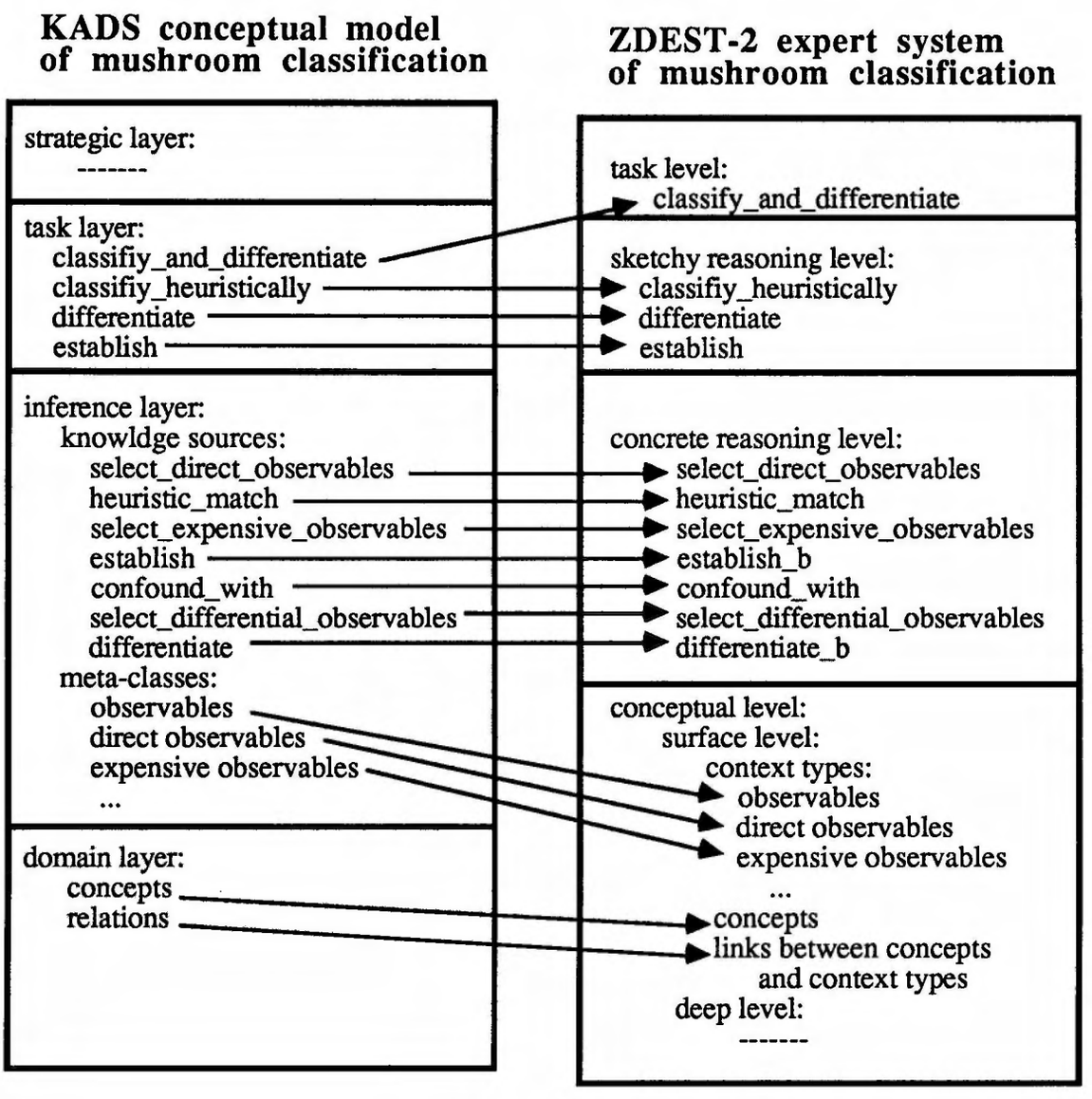


Fig. 5.1: map from conceptual model to ZDEST-2 architecture

Task level in ZDEST-2:

ZDEST uses a fixed scheduler as a general control structure of the expert system generated, so the strategic knowledge of scheduling and the highest control knowledge as expressed in the topmost task classify_and_differentiate was represented as a ZDEST-2 task.

Sketchy reasoning level in ZDEST-2:

At this level, the control knowledge for pursuing individual goals is described. Hence we mapped all lower level tasks of the conceptual model to meta-knowledge sources in ZDEST-2.

Basic reasoning level in ZDEST-2:

Here the detailed method of pursuing each primitive subgoal is described. Correspondingly, we mapped all knowledge sources of the conceptual model to basic knowledge sources in ZDEST-2.

Conceptual levels in ZDEST-2:

Since there is no theoretical but only heuristical knowledge in mushroom classification, there is no need for a deep conceptual level. On the surface level, we represented all the static knowledge of the conceptual model in a semantic network mapping meta-classes to context types, KADS´concepts to ZDEST-2´s concepts, and relations to links.

(6) Implementation

To refine the global architecture, we had to fill in the definitions of all control components, and we had to specify the semantic network, which again was rather straightforward.

Tasks:

classify_and_differentiate
  preconditions: true
  MKSs used: (classify_heuristically, differentiate, establish)
  endaction: nil
  MKS-order: if (success classify_heuristically)
    and (success establish)
    and (success differentiate)
    then (print_solution)

Meta knowledge sources:

classify_heuristically
  BKSs used: (select_direct_observables, heuristic_match)
  control: (forward_chaining)
  meta-rules: if (success select_direct_observables)
    and (success heuristic_match)
    then success
  context-type: (set_of_hypotheses)

establish
  BKSs used: (select_expensive_observables, establish_b)
  control: (forward_chaining)
  meta-rules: if (success select_expensive_observables)
    and (success establish_b)
    then success
  context-type: (hypothesis)

differentiate
  BKSs used: (select_differential_observables, confound_with, differentiate_b)
  control: (forward_chaining)
  meta-rules: if (success confound_with)
    and (success select_differential_observables)
    and (success establish_b)
    then success
  context-type: (differential_hypothesis)

Basic knowledge sources:

Since all basic knowledge sources perform operations on the conceptual knowledge level, they essentially consist of procedure calls to extract this knowledge, intermediately store and retrieve it. Such procedures are built-in in

ZDEST-2. For the purposes of this paper, this description should be sufficient.

Semantic network:

SNL expressions like

```
($space mushroom_classification
 ($space mushrooms
       /boletus//
            ...)
 ($space observables
   ($space direct_observables
         /form_of_pileus (direct_observ boletus)//
          ...)
   ($space expensive_observables
        /taste (expensive_observ boletus)//
         ...)            )         )
```

generate a semantic network as sketched in figure 5.2:

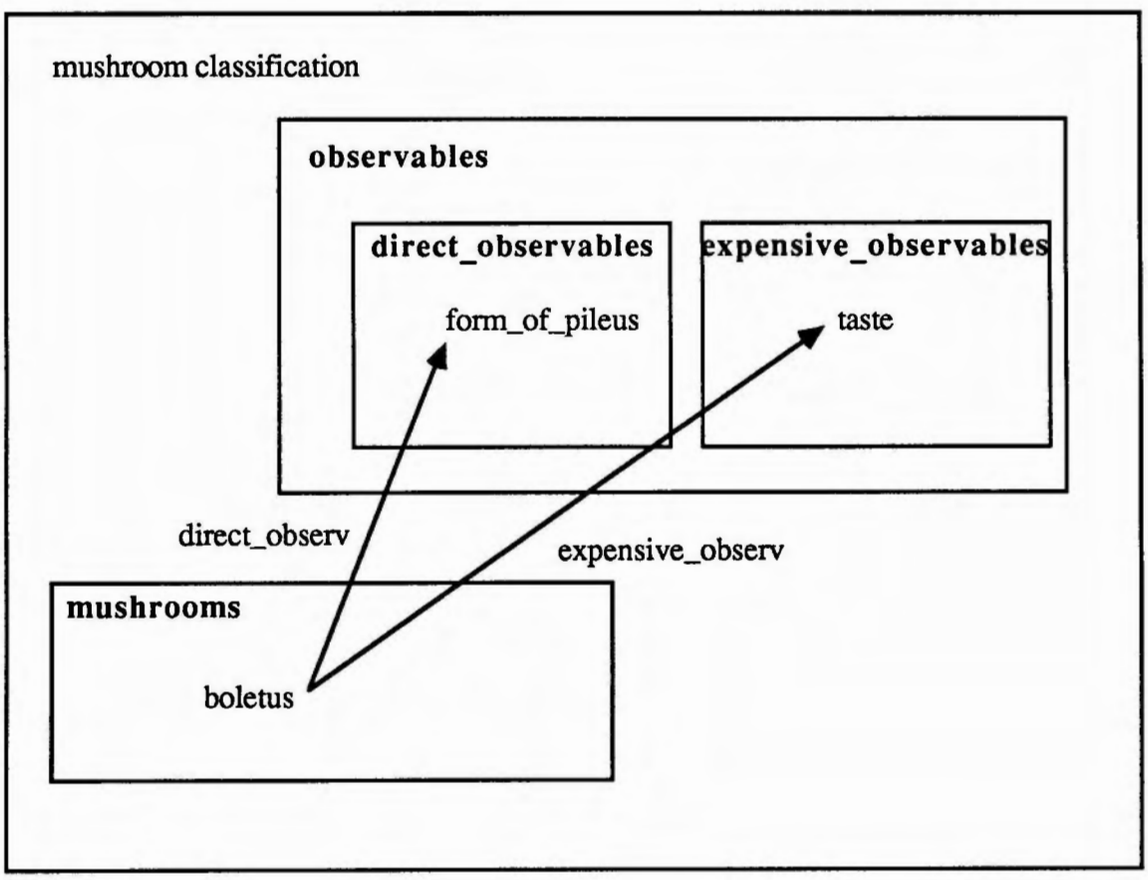


Fig. 5.2: SNL representation of the conceptual level for mushroom classification

6 Results

In our experiment, the transformation from the conceptual KADS-model to a ZDEST-2 expert system was surprisingly straightforward. In general, whith a non-trivial strategic

layer, it may be more complicated. Before we actually performed our experiment, we supposed that KADS strategies had to be mapped to ZDEST-2´s tasks as well as to meta knowledge sources, and that KADS´ tasks had to be mapped to meta- as well as basic knowledge sources in ZDEST-2. The rest of the mapping conformed with our expectations.

Figure 6.1 summarizes the approach we propose: It starts with linguistic data, from which a conceptual model is built according to the KADS methodology. Then a design description is constructed by defining the global architecture of the prospective system wrt. a high level expert system tool like ZDEST-2. It is refined to a detailed design description by filling in the definitions for all constructs declared in the architecture. Since the result is a complete, operational knowledge base, detailed design description and implementation coincide.

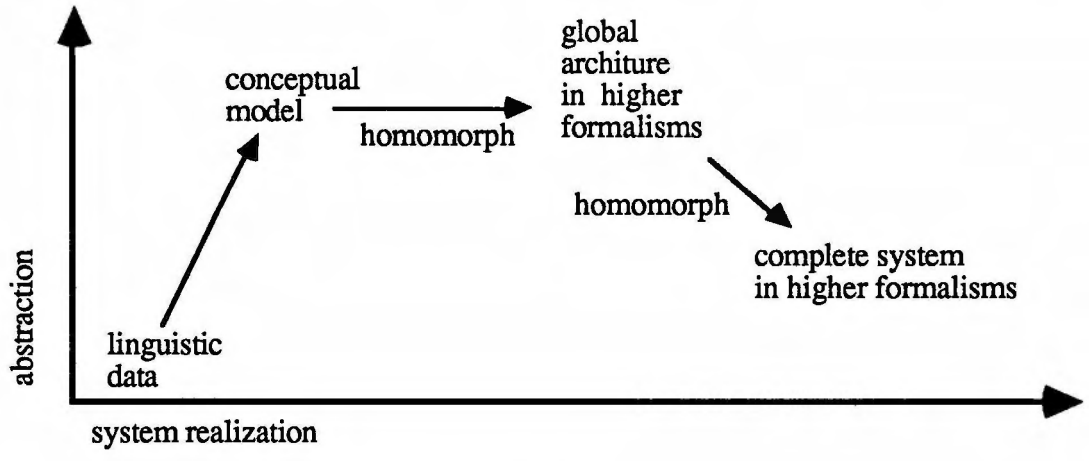


Fig. 6.1: our approach to knowledge acquisition

We see two problems in our approach:
Firstly, all aspects concerning user interfaces were excluded from our experiment. Here we think, that user interfaces and problem solver should be kept separated as far as possible, which can be achieved e.g. by using a blackboard interface between the two.
Secondly, we are aware that our approach will probably not yield very efficient expert systems.

On the other hand there are several advantages:
Firstly, our approach yield very transparent expert systems, which will outweight ineffiency when the human user has to understand the results of the expert system (because he has to account for it), or when the system will often have to be adapted to changing environments.
Secondly, the knowledge base of the operational system still contains epistemological categories like symptom or hypothesis, which is still umusuall in today´s expert systems.
Thirdly, conceptual models can be tested: It is far more easier for the expert to verify the representation of his expertise by watching the behavior of the model than by studying the paper documents of the model.
Fourthly, by separating the knowledge into layers, parts of it -like domain or problem solving knowledge- become reusable separately.

So we are rather pleased with the result. It encourages us to believe that we can build expert system tools on a higher level than is standard today that can be used to make conceptual models run while preserving their structure. We think this is important because conceptual models are comprehensible for experts and so should be their operational derivatives.

Literature

(Breuker, Wielinga 84) Breuker, J., Wielinga, B.: Techniques for Knowledge Elicitation and Analysis, Report 1.5 Esprit Project 12

(A1 report 87) Breuker, J., Wielinga, B., van Someren, M., de Hoog, R., Schreiber, G., de Greef, P., Bredeweg, B., Wielemaker, J., Billault, J-P.: Model-driven knowledge acquisition: interpretation models, task A1, Esprit Project 1098, Memo 87.

(Bennett 85) Bennett, J.: ROGET: a knowledge based system for acquiring the conceptual structure of a diagnostic expert system, Journal of Automated Reasoning, vol 1.

(Chandrasekaran 8) Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: high level building blocks for expert system design, IEEE Expert 1986.

(Gruber, Cohen 86) Gruber, T., Cohen, P.: Design for acquisition: principles of knowledge system design to facilitate knowledge acquisition, Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, 1986.

(Müller 86) Müller, B.: Lehrmaterialien BABYLON - Die Beispielwissensbasen zur Pilzbestimmung, Arbeitspapiere der GMD Nr. 221, 1986.

(Musen 86) Musen, M. Fagan, L. Combs, D., Shortliffe, E.: Using a doamin model to drive an interactive knowledge editiing tool, Proc. of the Knowledge Acquisition for KBS Workshop, Banff, Canada, 1986.

(Olson, Rueter 87) Olson, J., Rueter, H.: Extracting expertise from experts: Methods for knowledge acquisition, Expert Systems, August 1987

(Schreiber, Breuker, Bredeweg, Wielinga 88) Schreiber, G., Breuker, J., Bredeweg, B., Wielinga, B.: Modelling in KBdevelopment, to appear in Workshop Expert Systems and Applications, Avignon, 1988.

(Tong, He, Yu 87) Tong, X., He, Z., Yu, R.: A tool for building second generation expert systems, IJCAI 1987.

(Wielinga, Breuker 86) Wielinga, B., Breuker, J.: Models of expertise, ECAI 1987.

Collaborative Construction of an Office Knowledge Base: Experiences and Suggestions

F. von Martial, F. Victor
Gesellschaft für Mathematik und Datenverarbeitung (GMD)
P.O.B. 1240
5205 Sankt Augustin 1
Fed. Rep. of Germany

Abstract

We present our experiences in knowledge acquisition and representation we have made in modelling a medium sized organization. The resulting knowledge base is meant for different applications like information retrieval, advising and planning. An efficient knowledge acquisition has influence on the requirements for time and manpower, and the quality of the knowledge base. In our paper, we consider the coordination of a knowledge base construction team, the method of integrating common sense, the testing of the knowledge base and the selection of adequate knowledge representation formalisms.

1. Introduction

Knowledge representation and acquisition are active fields in office automation research. For restricted application domains, where the closed world assumption is valid, a lot of successful efforts have been made in constructing huge and hybrid knowledge bases, e.g. for technical expert systems.
The office world, however, is an open ended domain. The knowledge is unstructured, and social aspects have to be taken into account. Therefore, the knowledge acquisition process in the office domain is much harder than in most other fields. It is our objective to build an organizational knowledge base that models the essential aspects of the office world including common sense knowledge. In this paper, we describe our experiences in acquiring organizational knowledge and classify the problems we had into four main categories. A model for the collaborative construction of large knowledge bases is proposed that might reduce the impact of these problems on the efficiency of the knowledge acquisition process and the quality of the knowledge base.

We started out to model our own company, the GMD. The resulting knowledge base is meant for the use by office workers for information retrieval, advising and office procedure planning [MaVi88].
The characteristics of the office world have been studied in a lot of articles, e.g. in [ElNa87, Gibb85, HeJo82]. One result concerning the modelling of office knowledge is that the process of knowledge acquisition is very difficult to manage because this domain is highly indefinite and unstructured [Barb83, Crof87].

During our work we encountered four main problems which are crucial in constructing a large office knowledge base:
- the problem of coordinating a knowledge base construction team
- the problem of functionality of knowledge
- the problem of integrating common sense knowledge
- the problem of applying adequate knowledge representation formalisms

A team of knowledge engineers is needed to construct a large knowledge base. Their work has to be carefully planned and coordinated. We will describe a cooperation methodology enhancing the efficiency of the knowledge acquisition and representation process.
The second problem can be solved by underlaying the acquired knowledge with a requirement semantics, that specifies how the knowledge should be used. Controlling the quality of the represented knowledge is a permanent and necessary process during the acquisition.
The problem of exploding common sense knowledge for an organizational knowledge base is solved by a sophisticated control strategy to reduce the amount of common sense items in the knowledge base.
The knowledge representation problem is concerned with the question which formalisms are appropriate to represent certain knowledge, in our case office knowledge. The intended purpose of the knowledge base and its modularity and changeability play an important role in this connection.

2. Requirements for Coordinating the Construction Team

The building of a huge knowledge base requires a team of knowledge engineers. A team can accelerate the knowledge construction process. This is important, because otherwise there is the danger that the resulting knowledge base becomes obsolete before it is completed.

In our case the initial source of knowledge was the organizational manual of our company GMD [GBO85]. This extensive manual describes on more than 300 pages the structure and organizational rules of GMD. The underlying textual source of knowledge facilitates doing the acquisition process in parallel by assigning different parts of the manual to the team members. If the main knowledge source is an expert who has to be interviewed, i.e. the transmission of expertise is done linguistically, a large team is not so useful.

There are several problems which arise from having a team of knowledge engineers. A careful preparation to coordinate their activities is required without having too much organizational overhead. We performed several experiments within a group of two to five persons to identify the problems of collaborative knowledge acquisition and to develop a methodology for it. We selected an arbitrary chapter of the organizational manual of GMD. Then the team members had to model different chapters within a week's time. At first, there was no obligation concerning the choice of knowledge representation formalisms and the style of collaboration. Afterwards we tried out several regulations for collaboration and representation.

We had to cope with several difficulties concerning the cooperation during the knowledge base construction:

• Danger of multiple definitions of knowledge base objects
It often happened that the same object was defined by several team members. Sometimes they meant the same object but used a different name for it.
Example: Nearly all of our team members modelled the object "employee". They used different names: "person, colleague, member of GMD, secretary, researcher". This lead to multiple descriptions of the same object.
It is necessary to have a well organized information exchange between the members of the team. A list of synonyms of terms which appear in the manual and of further vocabulary should be kept. We have tried to reduce multiple definitions by keeping a documentation of the already represented frames and slots.

• Omission of domain objects
In contrast to the multiple definition of objects it also happened that some important objects of the manual had been forgotten to represent.
There should be a clear distribution of responsibilities among the team members. Different areas of text or topics should be assigned to different team members. This is sometimes difficult because often there do not exist clear boundaries between the areas which are to be represented.

• Different arrangements of knowledge
Some members described the knowledge about an item in one object. Others spread out their description of the same item over several objects. Some members tended to represent the knowledge in a flat structure while others used a lot of hierarchies.
The following example gives three possibilities to arrange the fact "The employee Gordon applies for leave of absence":

```
(frame Gordon
      :slots has-applied-for-leave :value yes
      .
      .
      .)
```

```
(frame application-form-for-leave
      :slots who :value Gordon
      .
      .
      .)

(frame application-form
      :slots type :value leave
            who :value Gordon
      .
      .
      .)
```

A method to achieve a common structuring is to agree on a skeleton of the structure (taxonomy) before a detailed description starts.

• Diverging opinions about the same object
The team members had different opinions about the same object. This results in different descriptions of these objects. The clarification of terminology often requires plenty of time for discussion.

Example: In this example we show how two team members modelled the object "employee". The produced frames contain different slots, although the team members meant the same object.

```
(frame employee                  (frame employee
:slots      name                 :slots  first name
            office                       name
            phone-no                     birthday
            age                          institute
            company)                     project
                                         education)
```

There should be a regulation which prevents from spending too much time for discussion.

• Diverging knowledge representation formalisms
There were great discrepancies concerning the use and preference of knowledge representation formalisms. Afterwards, this caused a great effort which was necessary to unify the the different formalisms. This problem is also treated separately in another chapter of this paper.

• Varying depth of modelling
There was a great variety in the granularity of the knowledge description. While some group members preferred a scarce description for "their" objects others used a lot of frames for their description. The latter included also additional common sense items for the representation of terms of the manual. The addition of common sense items is discussed in more detail in chapter 4.

Taking these aspects into account we will now propose a methodology for the collaborative construction of an office knowledge base. This concept will make the division of labor efficient, and will minimize the amount of overhead for coordination and discussion. The knowledge acquisition process is modelled as a cycle with several phases:

Phase 1: Intensive interaction and group discussions
The purpose of this phase is to establish a common terminology, a skeleton of the knowledge structure (taxonomy) and the assignment of labour. The basis of this phase is a topic that is treated by one or several chapters of the underlying manual. In order to build a common terminology during a meeting, the shared access to a meeting database is desirable. A tool which provides this facility is investigated in the WYSIWIS (= What You See Is What I See) approach [Stef86]. Phase 1 is characterized by a lot of face-to-face meetings. The support of this kind of communication activity has been explored in the project NICK [Bege86]. The knowledge acquisition tool KRITON which has been developed in our institute [Died87] supports the building of a taxonomy of a knowledge base.

The output of this phase is a skeleton for a knowledge base including terms that appear in the selected area of the manual, a list of synonyms for these terms, a taxonomy of these terms and a bunch of related common sense terms. This raw knowledge base is partitioned into several areas. Each of these areas will be worked out by a different member of the team.

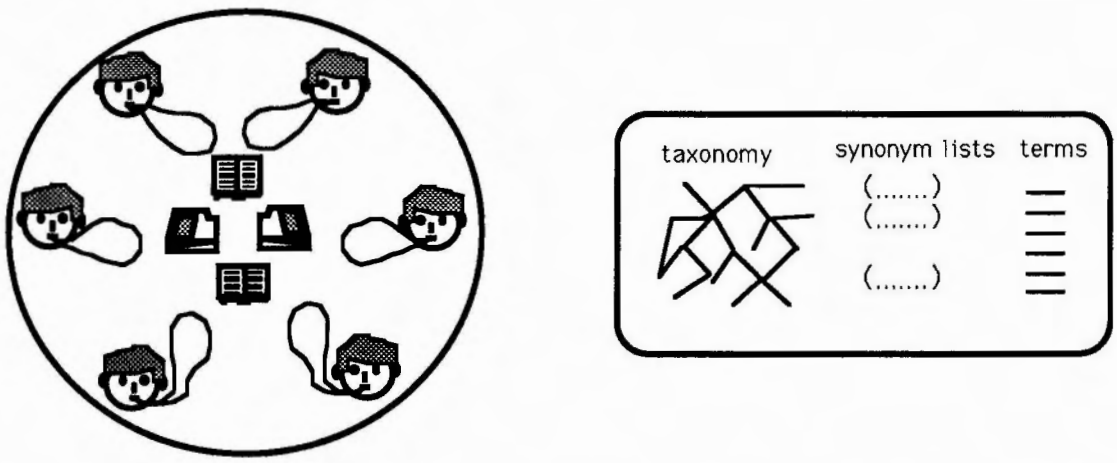


Figure 1: Phase 1 is characterized by a lot of discussions and delivers a rough outline of the knowledge base.

Example: The taxonomy concerning the topic "procurement" is shown in figure 2. In this case it is a hierarchy of purchase objects. The taxonomy is not complete and has to be refined during the next phase.

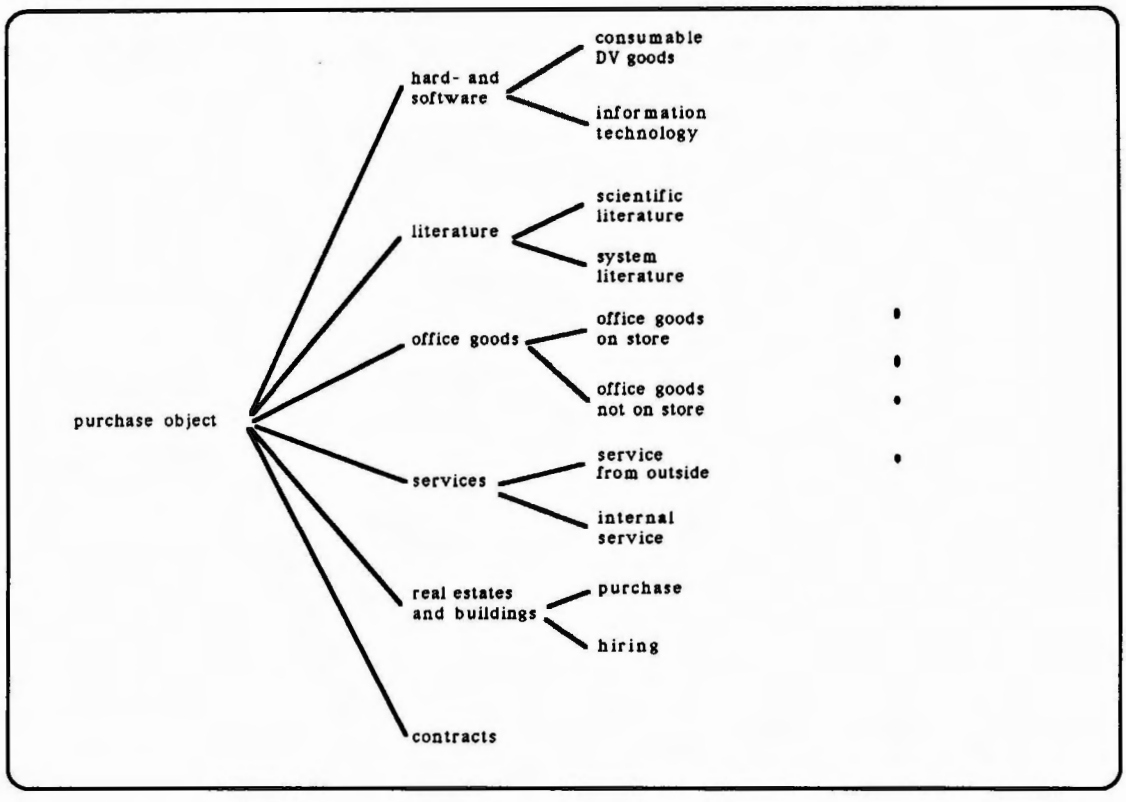


Figure 2: Extract of a purchase object hierarchy resulting from phase 1.

Phase 2: Separate work and person-to-person communication
During this phase every team member represents an area which has been outlined in phase 1. The objects which have been roughly sketched in phase 1 will now be described in detail. This also includes the integration of further common sense knowledge according to a strategy which will be described in the next chapter. It often happens that authors want to use terms from the already existing common knowledge base. In order to coordinate the access to this common knowledge base, one could employ techniques as described in [GrSa86]. It is also useful to get immediate information about objects which have been just defined by other authors, because an author may want to refer to this object. This prevents multiple definitions of the same object. Every team member should keep a list of topics which e.g. contains requests for modification of objects which have been defined by others so as not to disturb the work of the other team members with every modification request. Thus, time for discussion can be saved.
The team members should follow conventions for augmenting the knowledge base which have been defined in advance. A tool which allows the definition and control of such conventions is e.g. TINT [Nech86].
Tests concerning the functionality of the knowledge base will be performed according to ideas which are presented in the following chapter.
During this phase no group meeting takes place but there might be several person-to-person communications.

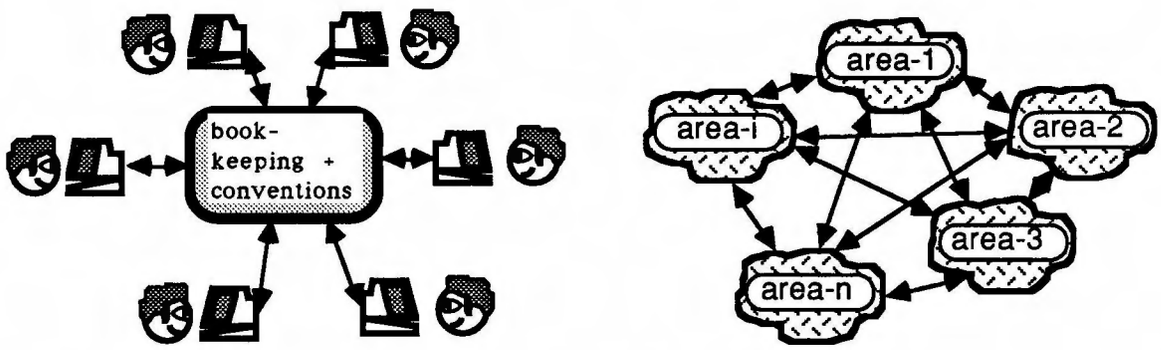


Figure 3: During Phase 2 the team members work on separate areas

Phase 3: Integration and discussion of open problems
The task of this phase is to integrate the different knowledge areas into the common knowledge base. Problems concerning the whole team, e.g. input conventions and terminology, will be discussed. The result of this phase should be a well-integrated and consistent knowledge base. After this phase has been completed phase 1 is entered again. The next run will increment the already existing knowledge base.

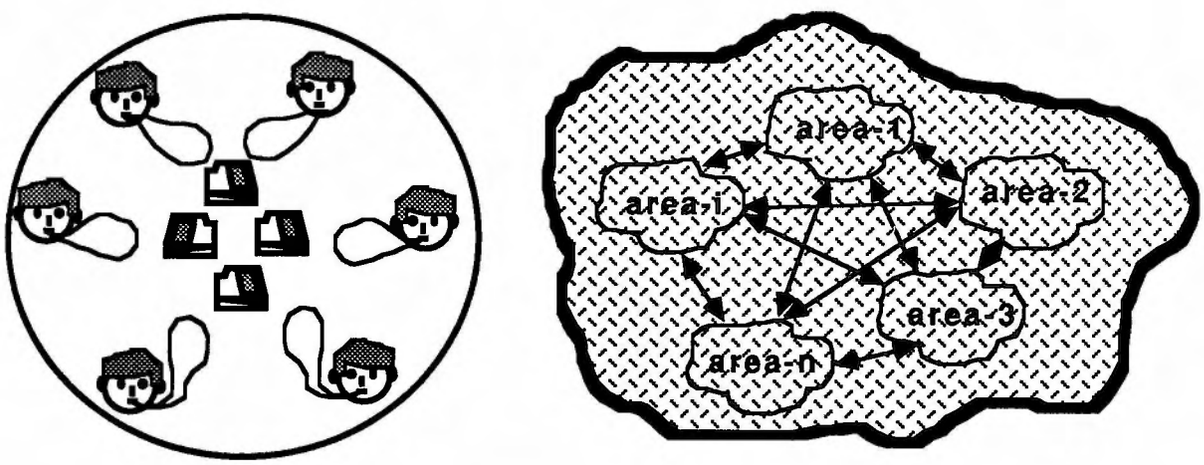


Figure 4: Discussion is necessary to integrate the areas into one consistent knowledge base.

3. The Knowledge Quality Problem

During our work on knowledge acquisition we have made the experience that it is not meaningful to represent knowledge without specifying the tasks that are to be performed with it. In [ByCh86] this principle is called the "interaction problem" and it is stressed that knowledge representation is always dependent on its use. Therefore, it is necessary to specify the functionality of knowledge before representing it.
In our case, we established a list of questions for the topic under consideration. Later, these questions should be answerable by using the knowledge base. One problem is how information can be retrieved from the knowledge

base in order to answer these questions. We use Prolog queries in addition to meta-reasoning concepts similar to the semantic matching algorithm in [KjCo87].
The retrieval mechanism has to take into account whether the closed world assumption is valid for the knowledge base, i.e. anything not known to be true is assumed to be false [ElNa87, HeJo82]. A consequence of this assumption is that the answer "no" for a retrieval might mean "false" in the sense of predicate calculus or "unknown", i.e. the information to answer the question is not represented in the knowledge base. In order to solve this problem, we want to integrate knowledge about the boundaries of the knowledge base as it is proposed in [Hayw86, Leve84].

The cycle of formulating and testing requirements for the knowledge base supports the control of progress in knowledge acquisition in an early phase of system development. This may reduce costs because too many knowledge base modifications in later phases of the development cycle are avoided.

Example:
We are applying the ideas mentioned above for building a large knowledge base containing organizational knowledge about our own company, the GMD. As our first topic we have chosen the "organizational hierarchy" since we assumed that this domain can be often referred to when modelling other topics. For this topic our list of requirements contains questions like the following ones:
- What are the projects in GMD?
- Who is working in institute F3?
- Where is Mr. Jung´s office and what is his telephone number?
- Who is the superior of Mr. Jung?

As a result from the test of these questions via Prolog predicates we often had to modify the knowledge base because it was one of our goals to get information easily, i.e. without complicated programming of retrieval predicates. Also the quality of the answers determines whether the knowledge base is to be modified. Often the retrieved information is too detailed, not relevant or not sufficient.
As a second topic we chose the purchasing activity of GMD. Office procedures for purchasing depend on the objects that are to be purchased. So we decided to split this second topic into two parts: the purchase objects and the purchase procedures. Again we made lists of requirements and tried to verify them. We have made the experience that we often could use already represented knowledge of the first topic. The requirement lists were useful to clarify the semantics of knowledge represented for a specific topic, i.e. how the knowledge can be used.

4. Integration of Common Sense Knowledge

What is common sense ?
Common sense is that kind of knowledge which is not contained explicitly in a manual, e.g. an organizational manual, but which the reader needs in order to understand the text. In addition to naive understandings about concrete objects (people, office furniture, etc.), also conceptual objects (goals,

ideals, intentions) [Elli83] belong to this kind of knowledge. In this chapter we will concentrate on common sense knowledge for which the static description as a frame is adequate, in contrast to procedural common sense knowledge which can be better described by algorithms or rules.

Why common sense ?
The mere representation of terms that appear in the manual - we will call them "original objects" - is too flat. It leads to knowledge "islands". Additional common sense facts are necessary to link these isolated pieces of knowledge. The links increase the degree of connectivity of a knowledge base. They are useful for context switching, reasoning, planning or intelligent information retrieval etc.

A not yet solved problem is to find a consistent concept of how to integrate common sense in a knowledge base and to which extent.

The CYC project [Lena86] is an attempt to develop a huge common sense knowledge base. The knowledge source for this project is the Encyclopedia Britannica. For every object that is put into the knowledge base all common sense facts for this object are also integrated. This strategy is applied recursively to the common sense items just entered. For the modelling of an indefinite domain like the office world this exhaustive strategy is not useful, because it leads too far from the original domain.

CYC´s brute forth approach to integrate common sense requires a long-term project and a large amount of manpower. The problem for smaller projects like ours with less manpower is how to reduce the expense of common sense input.

The restriction of the amount of common sense items to be input into the knowledge base is possible along two dimensions:

• breadth: the number of common sense items directly associated with an original item.
Example: If a special building of GMD, "B9", is described by the number of rooms, its location, its name, its cost and its style of architecture, then the breadth of "B9" is five.

• depth: the maximum distance (= number of links) between the original item and an additional common sense item. In a semantic network this distance means the number of arcs that are required to travel from one object to another.
Example: A team member established a chain of common sense items starting with an original frame "Gordon" describing an employee of GMD: "Gordon" --> salary --> money --> account --> bank. The depth of the item "Gordon" is four.

In the CYC approach there is no bound for any of these dimensions. This leads to an explosion of knowledge.

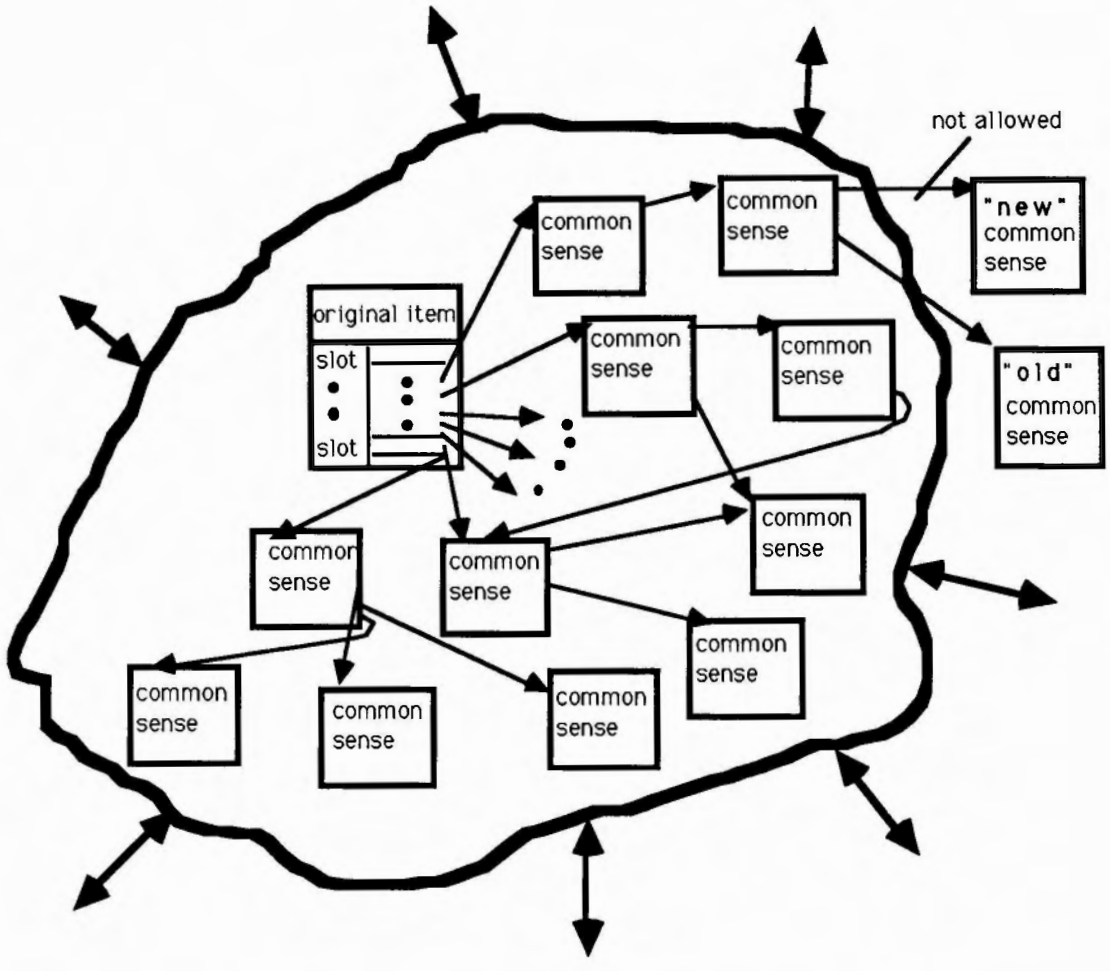


Figure 5: Strategy for a bounded integration of common sense knowledge

A restriction of the breadth does not save many additional items, instead it might lead to the pruning of important knowledge. However, a restriction of the depth for the expansion of knowledge is very suitable for the reduction of additional common sense items. It also assures that the items which are represented in the knowledge base are close to the modelled domain. After some experiments we found out an appropriate value of two for the maximum distance between original and additional knowledge objects. Based on this value, the amount of additional common sense items is controlled. If a distance of two is reached and a further reference to a nonexistent common sense object is desired, this reference should be replaced by a textual expression. But it is of course permitted to establish a connection over more than two steps if the referred object already exists. We have made the experience that this often is the case. During the process of knowledge representation the rate of additional common sense for an original object will drop as there is already a lot of common sense items in the knowledge base which can be referred to.

The control of common sense integration has a positive effect on the cooperation because it prevents the knowledge representation team from using too varying depths of modelling.

5. The Adequate Knowledge Representation

"Transforming knowledge into a form that can be executed by the computer is difficult because knowledge in the world is expressed in a different form than that required by the machine" [GrCo87]. This is also known as "representational mismatch" and it concerns our work as a main problem. Since knowledge in our application domain is of hybrid character and indefinite in nature [HeJo82, ElNa87], we felt the necessity to study and compare several representation formalisms according to their adequacy. We have been experimenting with a hybrid knowledge representation which includes frames, rules, procedures and nets. The structure of our knowledge base is described in [MaVi87].

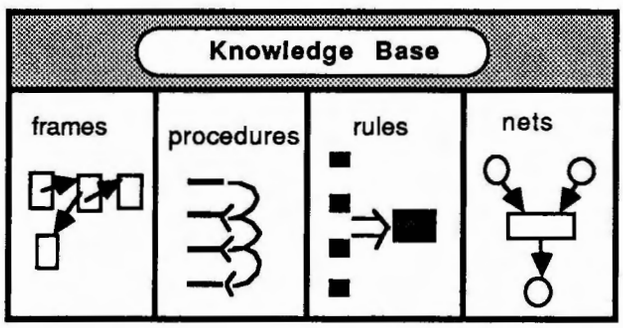


Figure 6: Several representation formalisms were used.

During our work we mostly had the problem of determining, what formalism should be used to model a specific piece of knowledge. We performed several experiments with rule based and object oriented concepts, e.g. we tried a rule based approach for modelling the organizational hierarchy of GMD. Our experience was that the represented knowledge was not well structured. It was difficult to change or update the knowledge base - a problem that daily occurs in organizations. So we decided to arrange the organizational hierarchy in an object oriented manner via a part-of hierarchy.

We have also made the experience that the modelling of procedural knowledge, e.g. office procedures, is much harder than the modelling of objects. Office procedures are characterized by a high degree of concurrency and division of labor, combined with handling different types of structured information. A net based approach [MaVi88] is well suited for the description of office procedures. Figure 7 shows a net for an office procedure "check-budget".

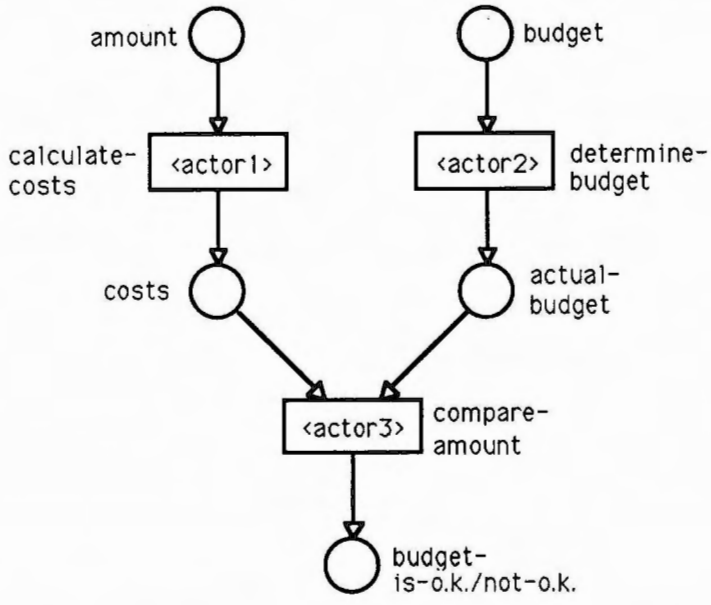


Figure 7: Representation of an office procedure.

6. Summary

Four main issues concerning the construction of a huge office knowledge base and their interrelationships have been pointed out. The underlying source of knowledge was textual. A central problem was to coordinate the acquisition and representation team. Our methodology for the performance of team work ensures a reduction of organizational overhead for coordination and discussion, a coverage of the application domain without doing multiple work, and a quick growth of the knowledge base. In order to guarantee the quality of the resulting knowledge base we have suggested quality checks to be integrated into the design process. We have emphasized the importance of common sense for a useful knowledge base. We had the problem of representing the right amount of common sense. To solve this problem we developed a strategy which controls the amount of additional common sense items by a bound.

Acknowledgements

We would like to thank Peter Wisskirchen for encouraging us to do this research, Thomas Kreifelts for his valuable comments and the WISDOM team of GMD for their cooperation.

References:

[Barb83] G. Barber, P. de Jong, C. Hewitt, "Semantic Support for Work in Organizations", in: "Information Processing 83", R.E.A. Mason (ed.), Elsevier Science Publishers B.V., North-Holland, 1983

[Brac86] R.J. Brachman, "I Lied about the Trees: Or, Defaults and Definitions in Knowledge Representation", The AI Magazine, pp. 80-93, Fall 1986

[Bege86] M. Begeman, P. Cook, C. Ellis, M. Graf, G. Rein, T. Smith, "Project Nick: Meetings Augmentation and Analysis", in: Proc. of CSCW´86 - Conference on Computer-supported Cooperative Work, Texas, 1986

[BeKa86] G. Berry-Rogghe, R.M. Kaplan, "On the Structure of Expository Texts; Preliminaries to Building an Expert System from Manuals", in: "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada, Nov. 1986

[ByCh86] T. Bylander, B. Chandrasekaran, "Generic Tasks for Knowledge Based Reasoning: The Right Level of Abstraction for Knowledge Acquisition", in: "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada, Nov. 1986

[Crof87] W.B. Croft, "Representing Office Work with Goals and Constraints", Proc. of the IFIP WG.8.4, Workshop on Office Knowledge; Toronto, Canada, August 1987

[Died87] J. Diederich, "Knowledge-based Knowledge Elicitation", in: Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milano, 1987

[Elli83] C.A. Ellis, "Formal and Informal Models of Office Activity", in: Information Processing 83, R.E.A. Mason (ed.), IFIP, 1983

[ElNa87] C.A. Ellis, N. Naffah, "Design of Office Information Systems", Surveys in Computer Science, Springer, 1987

[GBO85] "Geschäfts- und Betriebsordnung (GBO)" der Gesellschaft für Mathematik und Datenverarbeitung mbH, 1985

[Gibb85] S.J. Gibbs, "Conceptual Modelling and Office Information Systems", in: "Office Automation: Concepts and Tools", D. Tsichritzis (ed.), Springer, 1985

[GrCo87] Th. Gruber, P. Cohen, "Principles of Design for Knowledge Acquisition", Proc. of the Third Conference on Artificial Intelligence, Applications, Orlando, 1987

[GrSa86] I. Greif, S. Sarin, "Data Sharing in Group Work", in: Proc. of CSCW´86 - Conference on Computer-supported Cooperative Work, Texas, 1986

[Hayw86] S.A. Hayward, B.J. Wielinga, J.A. Breuker, "Structured Analysis of Knowledge", in: "Knowledge Acquisition for Knowledge-Based Systems Workshop", Banff, Canada, Nov. 1986

[HeJo82] C. Hewitt, P. de Jong, "Open Systems", Artificial Intelligence Laboratory, MIT, December 1982

[KjCo87] R. Kjeldsen, P. Cohen, "The Evolution and Performance of the Grant System", Proc. of the Third Conference on Artificial Intelligence Applications, Orlando, 1987

[KrSe87] Th. Kreifelts, P. Seuffert, "Application Scenario Procurement: The Procurement at GMD and its Modelling in the DOMINO-system" (in German), WISDOM Research Report FB-GMD-87-09, April 1987

[Lena83] D. Lenat, A. Borning, D. McDonald, C. Taylor, S. Weyer, "Knoesphere: Building Expert Systems with Encyclopedic Knowledge", Proc. of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West-Germany, 1983

[Lena86] D. Lenat, M. Prakash, M. Shepherd, "CYK: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks", AI Magazine, pp. 65-85, vol. 6, Winter 1986

[Leve84] H.J. Levesque, "The Logic of Incomplete Knowledge Bases", in: "On Conceptional Modelling", M.L. Brodie, J. Mylopoulos, J.W. Schmidt (eds.), Springer 1984

[MaVi87] F. von Martial, F. Victor, "The Electronic Organizational Handbook: Requirements and Specification" (in German), WISDOM Research Report FB-GMD-87-16, July 1987

[MaVi88] F. von Martial, F. Victor, "An Interactive Planner for Open Systems", Proc. of the Fourth IEEE-Conference on Artificial Intelligence Applications", San Diego, 1988

[McDa87] J.P. McDowell, K. Dahlgren, "Commonsense Reasoning with Verbs", in: Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milano, 1987

[Nech86] R. Neches, "Tools Help People Co-operate Only to the Extent that they Help Them Share Goals and Terminology", in: Proc. of CSCW´86 - Conference on Computer-supported Cooperative Work, Texas, 1986

[Stef86] M. Stefik, D.G. Bobrow, S. Lanning, D. Tatar, G. Foster, "Wysiwis Revised: Early Experiences with Multi-User Interfaces", in: Proc. of CSCW´86 - Conference on Computer-supported Cooperative Work, Texas, 1986

Knowledge Acquisition from Text-based Think-Aloud Protocols: Situational Specifications for a Legal Expert System

Thomas Wetter ♦ and Franz Schmalhofer ▲

IBM Germany Heidelberg Scientific Center (♦)
Wilckensstr. 1a
D-6900 Heidelberg
☎ (+49) (-)6221 404212
EARN: WETTER at DHDIBM1

and

Cognitive Sciences Centre/McGill University (▲)
3700 McTavish Str.
Montreal, PQ IY2 H3A
☎ (001) 514 398 4256
BITNET: IN3N at MCGILLB
and
Psychologisches Institut, Universität Freiburg (▲)
Niemensstr. 11
D-7800 Freiburg

Abstract

This research attempts to elicit the knowledge which is not directly stated in a text but nevertheless required for the text to be understood. When the information or expertise communicated by a text is to be represented in an expert system (XPS), such situational knowledge must first be made explicit and represented in appropriate terms. The elicitation of situational knowledge, which was required for LEX1, a legal expert system developed in the LEX project, ([Haft87]) was performed by having subjects think out loud, while and after reading texts. Three different presentation procedures and different instructions were used. They were designed

- as to increase the number of verbalizations per text,
- so that the verbalizations would be cast in preselected terms (e.g. terms of a particular XPS shell) at a certain level of detail, and
- so that the verbalizations were rather independent of the local processing of the text itself.

Think out loud protocols of approximately 20 subjects were collected with each procedure and situational facts and instantiated rules were obtained. The results show which of the three procedures is best suited for text-based elicitation of general situation rules. Based upon these results a multi component procedure for knowledge acquisition for XPS is outlined.

Introduction

> *... As the accused drove on the state highway, another traffic participant turned from road d into the state highway. The accused swerved out of the way of the other traffic participant and hit the road delimiters. The damage to the road delimiters was valued at 200 marks. The other traffic participant left ....*
> *... After waiting for a while, the accused drove home.*

Lawyers who concern themselves with the described case will not only utilize law knowledge but in addition must activate a possibly quite large body of commonsense knowledge, which allows a complete situational interpretation of the text in terms of its characters, events, the characters' goals and motives, as well as temporal and causal connections between events and goals ([Schan77]). Such a situational interpretation is a prerequisite for the legal assessment of the case. For example it must be examined whether the described event constitutes an accident, which also depends upon the motives of its participants.

Artificial Intelligence research addresses the question of how similar knowledge processing may be performed with an XPS and how such a system can be effectively constructed. Presumably, this can be accomplished by solving the following two related design problems:

- For the selected domain the most appropriate knowledge representation (KR) scheme(s), and, closely related, inference mechanism(s) must be found. The solution of this design problem defines an expert system shell.
- The appropriate and a sufficient number of knowledge items, which can be filled into the developed KR scheme must be constructed, so that a domain model emerges.

Although it has been argued that system building and knowledge acquisition interact inseparably ([Bucha83]) the construction of an expert system may initially be approached to a lower or higher degree in a modular way.

The LEX Project

The LEX project is concerned with the development of a legal expert system, which can assess the legal implications of criminal law case descriptions such as the presented text about a traffic accident. Legal judgement or consultation in the civil law system consists of deducing one or more conditions of the (presumably offended) norm (paragraph) from facts found in the case description. Law application in a legal expert system hence consists of proving norm conditions (playing the role of theorems) from case facts (playing the roles of axioms) by means of rules defining the relations between the involved terms. By utilizing commonsense knowledge the system would thus have to prove from the above text that the accused was involved in an accident and that after waiting for a while he left the scene of the accident. The so obtained

conclusions may then be matched to the legal rules, whereby a legal assessment of the case should be obtained.

In the LEX project 1st order predicate logic (1stPL), (based on discourse representation theory, [Kamp81]) is used as knowledge representation scheme, with a sequent calculus ([Schoe85]) as inference mechanism. 1stPL and sequent calculus thus specify the characteristics of the LEX expert system shell. Given the requirements of German law application this shell is well suited for providing an appropriate argumentation for legal cases.

In order to perform a legal assessment with the LEX shell three separate knowledge sources are needed. Each of these knowledge sources must be acquired for LEX1. One may consequently say that three different knowledge acquisition problems must be solved:

1. The text describing the particular case must be encoded into the system.
2. Situational knowledge must be acquired, so that satisfied or offended legal norm conditions can be proven from the text describing the case.
3. The relevant legal knowledge must be acquired.

[1] The case information has to be newly entered for every single case which the system has to process. Since this ***case information*** is provided by texts, LEX1 has been programmed to automatically build a propositional representation from texts ([Barne86; Guent86; Zoepp84]). For example, the sentence
The accused swerved out of the way of the other traffic participant ..
is thus automatically mapped onto the following propositions[2]

```
  accused (u2)
& traffic participant (u14)
& swerve out of the way (u2,u14,e4)
& time (t4,e4)
& before (t4,now).
```

Contrary to this repeatedly required encoding of case information, the situational and legal knowledge constitute the knowledge base of the expert system, which ideally is acquired once and for ever, and is general enough to process a number of different cases.

---

[1] In LEX1, the knowledge about a specific case is only temporally encoded. Applying the term knowledge acquisition for the encoding of the case description is therefore inappropriate in the strict sense. The methods of automatically encoding information from texts could, however, be extended to the incremental encoding of text information, which would then constitute a knowledge acquisition process from written documentation. An additional approach to knowledge acquisition from natural language sources is discussed in [Wette88].

[2] To give a clear account of the central issues of the presented approach, the linguistic detail of the German implementation had to be significantly simplified in some places.

The ***legal knowledge*** base about hit and run driving, has been developed in a more or less conventional knowledge editing fashion in close cooperation of a knowledge engineer with a domain expert.

This investigation deals with the acquisition of the corresponding ***situation knowledge.*** Although the think aloud procedures were only used to elicit commonsense situation knowledge, another important perspective of this research lies in the development of a text-based procedure for acquiring situation and expert knowledge through thinking out loud.

The results of our study will therefore be used for suggesting a general knowledge acquisition procedure.

Knowledge Acquisition Methods

A number of different procedures have been discussed which have been subdivided into two groups, namely knowledge editing (cf. [Musen87] as a typical example) and knowledge acquisition techniques (cf. [Boose87]as a typical example).[3]

The differences and the characteristics of these methods determine the kind and the amount of knowledge which they elicit from a human subject. In the case of ***knowledge editing*** the expert system shell determines the concepts and the structure which are predominantly available for coding knowledge. Thereby, the expert is guided to provide that information which can be expressed within the existing structure. He must also adhere to a minimal set of syntactic conventions such as quoting both premise and conclusion of a rule. The knowledge acquisition according to the principles of the ***personal construct theory*** (cf. [Boose87]) is independent of the syntax of the target language. Here, the acquisition process is affected by the object descriptions which the human expert has previously named and which are entered into the grid. In cases where a solid domain theory exists asking such very direct questions may be quite appropriate and particularly useful for obtaining declarative knowledge. If, however, such domain knowledge is not available, unjustified presuppositions in the question may yield incorrect answers. Questions with unwarranted presuppositions may thus affect the subject's answer in an intrusive way. Therefore, knowledge acquisition systems such as AQUINAS ([Boose87]) seem to be particularly suited for the elicitation and hierarchical classification of declarative knowledge items in the presence of a solid domain theory.

Instead of asking detailed questions or imposing very specific task demands in the knowledge acquisition, ***this approach*** attempts to record the information which humans actually use for comprehending a text. When subjects read a text the relevant situation knowledge is presumably activated in their memory. Under think aloud instruction this activated information would at least partially be verbalized. What is being verbalized depends upon the instructions given

---

[3] Machine learning, which forms an important approach by itself, is not further considered here.

to the subjects as well as on the particular way by which the text is presented. [Erics84] showed that under concurrent think-aloud instructions, humans verbalize the information which s being heeded in working memory during the performance of the task. In c text comprehension task, one can thus expect the verbalization of inferences ([Olson81; Guind81]) and the default knowledge which is used for comprehending the specific text. Thereby, subjects usually do not verbalize general entities such as complete procedures or rules but only their task specific situations.

The Elicitation of Situation Knowledge

In a more general sense, situation knowledge can be defined as any domain specific knowledge which is required in addition to linguistic knowledge for understanding a text. Expert knowledge in medicine, engineering or chemistry would thus form the situation knowledge for understanding the professional texts of these domains. The understanding of texts about traffic accidents requires commonsense knowledge rather than such highly specific expert knowledge. For different texts expert and commonsense knowledge may thus serve the same function. Whereas there are relatively few experts for highly specialized domains, [Hayes85]has pointed out that almost everybody is an expert of the commonsense world. Depending upon the topic of the text, a smaller or larger set of persons thus have the relevant situation knowledge. The understanding of texts about traffic accidents requires commonsense rather than specialized expert knowledge, so that for those texts almost everybody has the relevant situation knowledge.

Variants of the Think-aloud Method

Three variations of the think-aloud method are presented. Their adequacy for eliciting situation knowledge was investigated with samples of subjects on the commonsense domain of traffic scenes. All methods need post-processes to eliminate redundant and irrelevant statements from the protocols so that the information of interest remains and can subsequently be coded for the use in the expert system. The different verbalization procedures were designed

- so as to increase the number of verbalizations per text,
- so that the verbalizations would be cast in preselected terms and at a certain level of detail, and
- so that the verbalizations were performed after a scenario had been constructed from the text.

Concurrent Think-aloud Procedure (CTA) The text is divided into segments so that each segment corresponds roughly to a propositional unit. For example the previously stated sentence-segment is divided into four parts (indicated by <>):

The accused swerved out of the way <>
of the other traffic participant <>
and hit <>
the road delimiters.

The text segments are sequentially presented on the computer screen. The subjects are instructed to read out loud each segment and verbalize all their emerging thoughts. After pressing a button the next text segment appears on the screen directly to the right side of the preceding text segment. If there is not enough space in the current line, a new line is started. Since the preceding text segments remain on the screen the subjects can always see all previous text segments. The presentation of the text segments and the recording of the subjects' verbalization are synchronized.

Concurrent Scene Refinement (CSR) For the concurrent scene refinement larger text segments, roughly corresponding to the clause of a sentence are again presented in sequence on a computer screen. After finishing the processing of a text segment the subject presses a button and the next text segment appears on the screen. Subjects are instructed to find that information which is not explicitly stated in the text, but which is needed as background knowledge for understanding the current and the previous sentences of the text. In order to clarify these instructions a specific example of the to-be-found situation information is presented for an example sentence beforehand. Subjects are instructed to verbalize that situation information for each text segment. For assistance they receive a short list of words, which appear useful for the task, but they are free to use any other words they prefer.

Scenario Construction (SC) According to this procedure subjects first have to read the text aloud and design a map on a large sheet of paper of the situation which is described by the text. Thereafter the temporal and causal sequence of events is to be enacted on this map by moving toy objects corresponding to the objects described in the texts to the respective locations on the map according to the temporal sequence described or implied by the text. Thereafter the text is removed from the subject and the subject has to produce a scenario of the temporal and spatial relations between events and objects. This description is orally stated and recorded by a dictaphone.

Collecting Verbal Protocols
The concurrent think-aloud procedure was tested with twenty students from the University of Freiburg. Five different texts were presented in sequence. Thereafter the scenario construction procedure was applied with the same twenty subjects and the identical texts. Seventeen new students from the University of Freiburg were employed for using the concurrent scene refinement, again with the same five texts.

Knowledge Classification

The think-aloud protocols obtained in the three tasks are further processed in a semi-formal way. The same analysis is applied for all three procedures. Each subject's verbalizations are first divided into segments. For a more detailed description of this segmentation see [Schm88a] . Each segment is then classified as belonging to one of the different categories shown in Table 1 .

Table 1. Example verbalization for each category

category	text fragment	verbalization		condition
		translated	German source	
situation information				
central - fact	*The accused avoided ... traffic participant.*	*He saw the other traffic participant.*	*Er sah den anderen Verkehrsteilnehmer.*	*CRS*
central - rule	*and hit the road delimiters.*	*Because of the swifting out of the way, he hit the road delimiters.*	*Wegen dem Ausweichmanöver prallte er gegen die Leitplanken.*	*CRS*
peripheral	*As the accused drove on the state highway,*	*He is not in prison.*	*Er ist nicht im Gefängnis.*	*CSR*
text information				
paraphrase	*and hit the road delimiters.*	*He did not only lose control but even hit the road delimiters.*	*... verliert nicht nur ... Beherrschung ..., sondern er knallt auch noch gegen die Leitplanke.*	*CRS*
meta-statement	*and hit the road delimiters.*	*The text didn't tell yet, what happened to the driver.*	*Was mit dem Fahrer geschah, wurde bis jetzt noch nicht beschrieben.*	*CSR*
Rest				
forward inference	*and hit*	*a parking car?*	*gegen ein parkendes Auto?*	*CTA*
other association	*valued at 200 marks.*	*A damage of 200 marks is only reported in the newspaper, when an important person was involved.*	*Schadensmeldungen von 200 DM stehen normalerweise nur im Lokalteil der Zeitung, wenn es sich um wichtige Personen handelt.*	*CTA*

For the purpose of the present study we were only interested in a certain subset of situation information. With respect to our knowledge acquisition task spatial, temporal, and motivational relations at the scene of the accident and in the close surrounding were of interest. Thereby scene relevant statements, which were further analyzed, were distinguished from only peripherally relevant statements.

Depending upon the contents and structure of the text, the reader may or may not perceive the same topics as relevant. It may be assumed that the script or schema which a reader selects for comprehending a text ([Bower79]) will determine the relevance of a statement. We had selected one text (text 3) with an accident which involved a prostitute who was being harassed. It was expected that subjects would verbalize more information about the harassment situation, which was classified as only peripherally relevant by our criteria. As can be seen from Table 2, this expectation was substantiated in the CTA condition.

There was only little variation in the verbalizations among the other four texts. Therefore, pooled average verbalization frequencies are reported for these four texts, separate from the verbalization frequencies of text 3.

	situation information			text information		***Rest***		total no. of verbalizations	total no. of texts
	central		peripheral	verbatim reproduction of text	meta-statements	forward inferences	other associations		
	fact	rule							
	(for texts 1, 2, 4, and 5)								
CTA	26.0	1.5	3.0	1.5	16.1	29.7	22.2	2282	68
CSR	78.9	2.9	3.2	5.2	5.4	1.9	2.7	2760	66
SC	81.5	0.8	1.5	11.1	1.5	0.6	3.3	690	37
	(for text 3)								
CTA	26.9	1.6	9.7	2.1	9.3	28.5	21.9	680	17
CSR	77.2	1.7	3.5	5.9	4.0	5.3	2.4	891	16
SC	72.7	1.0	7.4	14.9	1.5	0.8	1.7	403	18

Table 2. Percentage of verbalizations for the different verbalization categories: Concurrent think-aloud procedure (CTA), concurrent scene refinement (CSR), and scenario construction (SC)

Overall there was a slightly larger number of verbalizations in the CSR than in the CTA condition, and significantly less verbalizations in the SC condition than

in those two conditions. The relative frequencies for the different types of verbalizations are shown in Table 2 . The reported percentage scores show that in the CTA condition the maximum number of verbalizations (approximately 50%) occurs in the ***rest*** category. In the CSR and SC conditions, on the other hand, the situation information category contains the maximum number of verbalizations (approximately 75%). The reported frequencies of the SC condition are, however, somewhat misleading: in the SC condition the subjects' protocols could not be directly compared with a segment of the text. The classification of the SC protocol are, therefore, not quite comparable to the other two protocols. In particular, it should be noted that for the SC protocols some of the situation relevant facts would be classified under paraphrases, if a stricter criterion for situation relevant facts would be used. For the purpose of this investigation, the CSR protocols are therefore the most useful data. It is obvious that even in these protocols subjects did not verbalize rules but only facts, which was to be expected from psychological studies on protocol analysis (¢Erics84ö).

Different types of knowledge were verbalized in the three tasks. The CTA methods resulted in a number of forward inferences from the text, which either were or were not substantiated by later sentences of the text. These forward inferences may be mainly caused by the sequential presentation of very small text segments. An explanation for the large number of unjustified forward inferences can easily be derived with incurrent theories of text comprehension (¢Kints88ö). For our current objective such forward inferences are only of limited use. The CSR provided the largest fraction of relevant situation knowledge. About three times as much useful information was obtained as compared to the other conditions.

Knowledge Acquisition

From the subjects' scene relevant situational verbalizations rules were to be derived. Although subjects verbalized only a few rules, the stated facts can be interpreted as the consequences of the application of a rule whose condition part matched one or several propositions of the encoded text. From the stated situational facts and appropriately selected text segments rules may thus be constructed. The verbalization *"He had the right of way."* which was stated when reading *"As the accused drove on the state highway"* yields the rule *"If the accused drove on the state highway, he had the right of way."*. The verbalization *"The other person violated the right of way of the accused"* similarly yields the rule *"If the accused drove on the state highway and another traffic participant turned from road d into the state highway, he violated the right of way of the accused."* The statement *"The two vehicles did not touch each other."* yielded *"If the accused swerved out of the way of the other traffic participant, the two vehicles did not touch each other"*. The statement *"The accused had to avoid the other car."* yields *"If another traffic participant turned from road d into the state highway, the accused had to avoid the other car."* Similarly, the rule *"If the accused swerved out of the way of the other traffic participant, then the accused saw the other traffic participant"* was formed. These rules were obtained from the most frequently stat-

verbalizations to the first three sentences of the sample text in the CSR condition.

Supposedly these are not the rules which were used by the readers but incomplete copies of their instantiations. By using machine learning techniques, the appropriate rules may be constructed by variabilizing constants and other generalizations. For example, an object hierarchy obtained by repertory grids (AQUINAS) may be used as generalization hierarchy for determining the appropriate level of generalization. Thereby the following more general rule fragments may be obtained:

```
drive_on (x,y) &
state_highway (y)
  -->
     has_right_of_way(x)

in (x,b) &
enter (y,b) &
...
  -->
     violate_right_of_way(y,x)

swerve_out_of_way (x,y)
  -->
     see (x,y)

in (x,y) &
enter (z,y) &
...
  -->
     obliged_to_avoid (x,z)
```

The periods in the rules indicate that the rules are still deficient in a different way: They are now over-general, i.e. there premises must subsequently enriched by conditions restricting them to reasonable models. The second rule, e.g. is certainly not true in arbitrary spaces, but in many streets.

The rules reflect the various domains of the commonsense world, naive legal rules, as well as physical and moral rules. Whereas machine learning procedures may help for inferring general rules from the constructed rule instantiations, it is more difficult to find the missing propositions in the condition part, which are required to make the rule more specific and thereby correct.

It is interesting to compare the verbalizations from the CTA and CSR. One first has to determine the effect of presenting short vs long propositions as trigger for verbalizations. One main difference seems to be that CTA produces many forward inferences, which might be recalled from slots of a script activated by the first proposition. CSR supplies a much higher percentage of verbalizations concerning the factual text information as a whole, instead of speculating on how the text might be continued. To interpret the difference, we must also take

into account that the CSR method supplies the subjects with a selection of words to use for verbalizations.

The effect of these words could be:

- that they trigger additional inferences producing new working memory contents, which can subsequently be verbalized; this would be rather intrusive as compared to allowing the subjects to make only those inferences he/she would make anyway.

or

- that they filter out working memory contents hard to express (and hence little related) with the offered words, with the effect that the closely related working memory contents are further activated and preferably verbalized; this would be less intrusive in the sense that it does not refill but only re-emphasizes working memory.

Since additional inferences can be expected to require additional time, one would expect for the first interpretation that verbalizations take considerably longer in CSR than in CTA. This is clearly not the case (see Table 3, last column.). As a matter of fact, the less intrusive filter-hypothesis is supported.

	length of verbalization per text in no. of words	length of verbalization period in minutes	no. of words per minute
CTA	271	7.60	35.07
CSR	437	11.79	36.24
SC	162		

Table 3. Amount and speed in the three test conditions

Discussion

For the knowledge acquisition of expert systems the analysis of verbal data is certainly indisposable. In order to supply the appropriate and a sufficient number of knowledge items, human experts are often interviewed, answer to questionnaires, think out loud during the processing of a task in the field of their expertise, or their texts are used as knowledge sources. The knowledge acquisition for CRACK ([Fisch88]) which is a system for designing kitchens is a recent example of a practical application of these knowledge elicitation methods.

As the field of knowledge acquisition becomes a more mature subdomain in expert system research, the virtues and drawbacks of the various knowledge elicitation procedures must be understood so that a standard knowledge acquisition procedure can be developed for the various types of knowledge, such as declarative and procedural knowledge. Rather than simply coding the knowledge which makes a system work for a single task, standard knowledge elicitation procedures should therefore be developed which allow for a more systematic reconstruction and formalization of a larger body of expertise.

Such a procedure is also required for collecting and analyzing verbal data in a way, so that procedural knowledge or rules can be acquired from these data. It is known that interviews very easily yield consistent explanations but that these explanations do often not correspond to the true knowledge which is applied in performing the respective task ([Nisbe77]). Depending upon how an interview is conducted, it may therefore not unveal the knowledge which an expert really uses when performing a task, but only result in some well phrased description. Interviews yield descriptions, which have been internally edited by the expert for consistency and superficial completeness rather than the knowledge which is actually used when performing a task.

While in some technical domains the edited explanations may be easily told apart from the real arguments, when the so acquired knowledge is utilized in the expert system shell, this may not be true for legal and other judgement tasks. For legal expert systems it is of critical importance that authentic domain knowledge is obtained. It is well known that concurrent think aloud is better suited for this purpose than interviews. When XPS are to be designed as cognitive models of human expertise ([Schma87]), think-aloud data may therefore be an extremely important source of information. The results of the present study show how useful think-aloud protocols may be obtained by the method of concurrent scene refinement. These protocols most frequently contained situational consequences from the application of rules.

By selecting the text segment which the reader processed before the statement of the conclusion, a condition segment could be identified. Together with the verbalized conclusion a condition action pair was thus defined. From these condition action pairs general rules can be constructed by variabilizing the particular instances in the condition action pair. The so constructed rule may very well correspond to the rule which the reader applied when processing the text. Most noticeably, the so constructed rules often do not contain the facts in their conditions which must be assumed as generally valid for the class of text under study and are therefore too general. It may not be easy to remedy this deficiency.

Another problem is to deal with rules, whose distinction between premise and conclusion may be controversal. The connection between *to see* and *to swerve out of the way* is an example. Such cases indicate that the preselection of a knowledge representation and inference mechanism may severely restrict the capability of the XPS to express such human inference mechanisms, which were not evident in the beginning of constructing the system. As has previously been pointed out knowledge acquisition must be seen as an iterative process ([Bucha83]) in which knowledge elicitation and formal representation are inseparably interwoven. And think aloud data can certainly not be used for replacing the work of a knowledge engineer. They may, however, be an invaluable source of information for making the work of a knowledge engineer a success. Think aloud protocols are best suited for obtaining a trace of the procedural knowledge which an expert actually applies when he performs some task such as comprehending a text or solving some problem. Because of their incompleteness protocols should be collected from a number of subjects with varying levels of expertise(cf. [Schm88b]). After the relevant knowledge has been coded by a knowledge engineer according to the method described in this paper this knowledge elicitation should be iter-

ated. In the second and further iterations the expert should perform more specific tasks for example processing a certain case rather than just comprehending a text. In the law domain this may be the establishment of the proof that the accused was a participant in the accident. These protocols can then be used for supplementing the previously coded rules.

"A common view in AI is that, while expertise is 'surface' knowledge and can be extracted by the expert system builders fairly easily, common sense knowledge is 'deeper', more firmly buries in native machinery, and that to extract it would be much more difficult" ([Hayes85, p31], see also [Wette87]).

A more complete set of rules should therefore be obtained when the suggested procedure is applied for a more specialized field of expertise rather than commonsense knowledge. Subject think aloud data seem to be the best source for the acquisition of procedural knowledge. The current research suggests that the procedural knowledge which is encoded by production rules in human memory may be elicited by having subjects think out loud during reading of text in their field of expertise. After rules are extracted from these verbal protocols, more specific tasks are required for supplementing this partial knowledge base. An iterative knowledge acquisition procedure is therefore suggested. While such an empirically oriented knowledge elicitation procedure does not necessarily simplify the work of the knowledge engineer it may well provide a more authentic general and complete knowledge base for a variety of tasks rather than only the knowledge with which a single task can be performed.

References

[Barne86] Barnett, B., Lehmann, Hu. and Zoeppritz, M., *A Word Database for Natural Language Processing,* 11th International Conference on Computational Linguistics, Proceedings, Bonn, Germany 435-440 (August 1986)

[Boose87] Boose, J.H. & Bradshaw, M.J. *Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems* Int. J. Man-Machine Studies **26**, 3 - 28 (1987)

[Bower79] Bower, G.H., Black, J.B., & Turner, T.J. *Scripts in memory for text* Cognitive Psychology **11**, 177-220 (1979)

[Bucha83] Buchanan, G. et al. *Constructing an expert system* in: Hayes-Roth, F. et al. (Eds.) Building expert Systems. Addison Wesley, Reading Ma. (1983)

[Erics84] Ericson, K.A. & Simon, H.A. *Protocol analysis: verbal reports as data* MIT Press, Cambridge Ma. (1984)

[Fisch88] Fischer, G. and Morch, A. *CRACK: A critiquing approach to cooperative kitchen design* in: Proceeding of Intelligence Tutoring Systems, 1.-3. June 1988, Montreal.

[Guent86] Günthner, F., Lehmann, Hu. and Schönfeld, W. *A Theory of Representing Knowledge* IBM Journal of Research and Development, Vol. 30, No. 1. 39-56 (1986)

[Guind81] Guindon, R. *Use of verbal reports to study inferences in text comprehension* Proceedings of the Cognitive Science Conference, University of California at Berkeley (1981)

[Haft87] Haft,F., Jones,R.P. & Wetter,Th. *A Natural Language Based Legal Expert System Project for Consultation and Tutoring - The LEX Project* ACM Baltimore (1987)

[Hayes85] Hayes, P.J. *The Second Naive Physics Manifesto* in: J.R.Hobbs & R.C.Moore(eds.): Formal Theories of the Commonsense World; Ablex, Norwood NJ (1985)

[Kamp81] Kamp, H. *A Theory of Truth and Semantic Representation"*, Groenendijk et al. (eds.), Formal Methods in the Study of Language. Mathematical Centre Tract, Amsterdam (1981). Reprint in: Groenendijk et al. (eds.), Truth, Representation and Information. GRASS 2, Dordrecht: Foris.

[Kints88] Kintsch, W. *The Role of Knowledge in Discourse Comprehension: A Construction - Integration Model* Psychological Review (1988)

[Musen87] Musen,M.A., Fagan,L.M., Combs,D.M. & Shortliffe,E.H. *Use of a domain model to drive an interactive knowledge-editing tool* Int. J. Man-Machine Studies **26**, 105 - 121 (1987)

[Nisbe77] Nisbett, R.E. Wilson, T.D. *Telling more than we can know: Verbal reports on mental processes* Psychological Review **84**, 231-259 (1977)

[Olson81] Olson, G.M., Mach, R.L., & Duffy, G. *Cognitive ascpects of genre* Poetics **10**, 283-315 (1981)

[Schan77] R.C.Schank & R.P.Abelson *Scripts, Plans, Goals, and Understanding* Erlbaum, Hillsdale NJ (1977)

[Schm88a] Schmalhofer, F. & Boschert, St. *Differences in Verbalizations during Knowledge Acquisition from Texts and Discovery Learning from Example Situations* Text, 1988 (in press)

[Schm88b] Schmalhofer, F. & Wetter, Th. *Kognitive Modellierung: Menschliche Wissensrepräsentationen und Verarbeitungsstrategien.* in: Richter, M.M. & Christaller, T.; Künstliche Intelligenz Frühjahrsschulen 86 und 87, Springer, Heidelberg (1988)

[Schma87] Schmalhofer, F. *Expert systems as cognitive tools for human decision making* in: Mumpower et al. (Eds.) Expert Judgement and Expert Systems. Heidelberg: Springer, 269-288 (1987)

[Schoe85] Schönfeld, W. *Prolog Extensions Based on Tableau Calculus* Proceedings of the 9th International Joint Conference on Artificial Intelligence, Vol. 2, Los Altos, USA, 730-732 (1985)

[Wette87] Wetter, Th. *Common Sense Knowledge in Expert Systems* in: Mumpower et al. (Eds.) Expert Judgement and Expert Systems. Heidelberg: Springer, 339-358 (1987)

[Wette88] Wetter, Th. *Towards knowledge base construction in natural language dialogue* (in preparation)

[Zoepp84] Zoeppritz, M. *Syntax for German in the User Specialty Languages System* Niemeyer, Tübingen (1984).

Acknowledgement
Gabi Schellhorn and Erik Farin conducted the experiments and collected the data at Freiburg University.
Rainer Goebel and Otto Kühn contributed additional insights about different theoretical views of the investigation and coded rules into logic formulae. We want to thank them for providing the ground upon which the presented results could be based.

ACQUIRING GRAPHICAL KNOW-HOW: AN APPRENTICESHIP MODEL

David L. Maulsby and Ian H. Witten

Knowledge Sciences Laboratory, Department of Computer Science
The University of Calgary, 2500 University Drive NW
Calgary, Canada T2N 1N4

Phone: (403) 220-7683
Email: ..!alberta!calgary!maulsby, ..!alberta!calgary!ian, ian@calgary.cdn

ABSTRACT

This paper studies the acquisition of procedural knowledge, or "know-how", from end-users in the domain of interactive graphics. In order to develop an open-ended system that is not restricted to any particular class of drawings, heavy emphasis is placed on the user interface. Experts (we call them simply "teachers") express procedures constructively, using any of the tools available in the interactive drawing environment. Well-structured procedures, including branches and loops, are inferred using a variety of weak generalization heuristics. The teacher's attention is concentrated on the system's perceptual and inferential shortcomings through a metaphorical apprentice called "Meta-Mouse". Its sensors are predominantly tactile, which forces teachers to make their constructions explicit. Meta-Mouse generalizes action sequences on the fly and eagerly carries out any actions it can predict. Theoretical support for the design comes from two sources: geometric phenomenology, which confirms that powerful problem-solving methods are associated with common-place spatial reasoning; and the fact that Meta-Mouse automatically imposes important "felicity conditions" on the teacher's demonstrations.

I INTRODUCTION

While knowledge acquisition research usually focuses on the extraction and formalization of high-grade knowledge for high-profile expert systems applications, there is an equally pressing need for systems which garner more mundane know-how from non-expert users to help them in their day-to-day work.

One domain where this is of particular interest is office work. Several experimental systems have been built which allow office tasks and procedures to be represented and executed automatically (eg. [Zisman 84], [Hammer 80], [Kunin 82], [Barber 83], [Croft 84]). These systems require an expert to codify knowledge and couch it in the formalized representation provided by the system. More germane to automated knowledge acquisition are methods for programming by example, which effectively permit the end-user to serve as expert (eg. [Attardi 82], [Halbert 84]; see [Witten 87] for a recent survey). As yet, such systems are rather primitive when viewed from the perspective of knowledge acquisition. One difficulty with research in this area is the highly varied nature of office activity; indeed, there is even controversy about the extent to which office work really does involve

executing procedures [Suchman 87]. A more narrowly constrained domain is the editing of loosely structured text [Nix 84]; this has been less popular probably because it is hard to quantify how common such tasks are in practice.

This paper studies the acquisition of procedural knowledge, or "know-how," from end-users in the domain of interactive graphics. While still an excellent example of human-computer cooperation requiring continual, richly-structured interchange between the partners, this constitutes a much more clearly defined domain than does general office work. Popular drawing programs (eg. MacDraw), which augment a software analog of drafting tools with extensive editing capabilities and rudimentary positional constraints, provide an ideal environment in which to investigate knowledge acquisition from non-expert users. These facilities enable casual users to master easily some of the mechanical skills of draftsmen and to concentrate more upon design. Moreover, they allow the composition of operations into re-usable structures, which, in principle, constitute knowledge that could be re-used. Current drawing systems, however, provide no way to manipulate this knowledge, and in practice users experience considerable frustration from having to repeat procedures again and again.

As an example, Figures 1-3 illustrate several tasks to be programmed within a MacDraw-style utility. The input picture is transformed into the output picture by constructive operations having both *ad hoc* and derived parameters. Most users could, after some contemplation, generate effective sequences of operations to accomplish these tasks. They do so by operationalizing and combining knowledge from diverse sources, ranging from spatial intuition through principles of Euclidean geometry to an understanding of everyday routines such as sorting. Our job is to devise a mechanism to acquire these procedures by observing the end-user at work, capturing them in a form general enough that they could be re-used in other similar situations. Programming by example is much better suited to the typical user of MacDraw than a declarative representation such as ThingLab [Borning 86] employs.

We view knowledge acquisition in this domain as tantamount to teaching. Unlike other researchers in machine learning, we stress the transitivity of the verb "teach" — there must be some organism that is being taught. The cornerstone of our approach is a suitable metaphor through which users can accurately conceptualize the target of the teaching operation. A user who has a clear idea of the capabilities of the learner will automatically adapt his teaching strategies to capitalize on its strengths and compensate for its weaknesses. The success of the metaphor will determine how well the user's teaching matches the student's ability. This strategy has been proposed for the office domain also [MacDonald 87].

Our metaphorical apprentice, called a Meta-Mouse, is a graphical robot that is taught by being led through a task. In operation, Meta-Mouse resembles Logo's celebrated Turtle [Papert 80], but it remembers what it has done and even induces looping and branching control structures. Meta-Mouse's body is essentially a moveable menu of commands (see Figure 4), with a grasper and a sensor for graphical objects. Like a good apprentice, Meta-Mouse takes over the task it is being taught as soon as possible, by predicting and performing the action sequences implied by a loop or conditional branch. Subsequently, the programming process becomes a matter of debugging by adding new conditional branches.

The procedures Meta-Mouse learns express both constant and variable (functional) spatial constraints. Our prototype Meta-Mouse, called Basil, is a very short-sighted resident of Flatland [Abbott 1884] and therefore incapable of analyzing large areas of the display, or of noticing invisible constraints such as a gap between two boxes. Instead, he learns about constraints by participating in their construction. His teacher uses the familiar tools provided by computer drawing systems. Basil can grasp and carry an object, such as a ruler, and knows when he or whatever he is carrying collides with an object. A positional constraint can be expressed as a sequence of contacts. Suppose, for example, that the pillars of an arch should be 2 cm apart. The teacher could instruct Basil to draw a 2-cm line out from the lower-right corner of the first pillar, then move the second pillar until its lower-left corner touches the right end of the line.

The Meta-Mouse derives from the entwined traditions of graphical programming and programming by example. The notion of a mouse icon/menu that tracks the real mouse in order to program it is a creative misinterpretation of the use of a mouse icon for specifying the functional bindings of mouse-buttons, as described in [Myers 86]. Geometric construction based on body-centered coordinates has become the mathematical discipline of Turtle Graphics [Abelson 80]. The use of traditional constructive techniques in graphics programming has been formalized as a language [Fuller 86], and been applied interactively [Bier 86]. Smith's Pygmalion [Smith 75] used a graphical representation of program structure, the box replacing the "( )" of LISP. Pygmalion learned sequential programs from traces but required that the user symbolically indicate conditional branches and loops. Meta-Mouse induces these structures. Halbert concluded that graphical programming by example without symbolic augmentation is practically impossible, and used data forms for the specification of variables and conditions [Halbert 84]. In our domain, where semantics and syntax are essentially identical (spatial relations), variables and conditions can be communicated without symbolic representation; variables, as the latent properties of a construction, and conditions, as visible events that precede or follow an operation. Andreae's robot programming system, NODDY [Andreae 85], most closely resembles ours, but performs general function induction to identify constraints; this would be impractical in our domain. NODDY employs a clever method of matching traces to procedures by propagating matches bi-directionally from key events. Hence, loops cannot be inferred from the first trace, which we consider a deficiency in our occupational environment.

In the next section we describe a simple graphical task, moving boxes to an arbitrary guideline, as it could be taught to Basil the Meta-Mouse. Two following sections discuss the acquisition of graphical procedures more generally; first with respect to the elements of graphical know-how, next by describing the Meta-Mouse in more detail. Finally, we describe the results of a small pilot experiment with potential users, and assess the requirements placed upon the teacher, and the means by which Basil helps the teacher meet them.

II BOX-TO-LINE: A WORKED EXAMPLE

Imagine being faced with the task of writing a program which, given a set of boxes, aligns their lower right-hand corners to an axis (guideline) given by the user at run-time. A trace is shown in Figure 6. The procedure is to translate each box horizontally to the guideline (the program will fail when the guideline is horizontal). First, suppose the task is performed *ad hoc,* without regard to programming. To permit exact alignment a visible

guideline is necessary; besides, it's easier to get the axis just as we want by drawing it explicitly. The guideline is made long enough that no box could pass horizontally without touching it. Having set up the task environment, the alignment operation is performed iteratively over the set of boxes. Each one is grabbed in turn and dragged horizontally (more or less) until its lower right corner is on the guideline. The task environment is then dismantled by removing the guideline.

Now consider this task as a program taught to Basil the Meta-Mouse. The teaching process consists of setting up the task environment, tracing the execution, and cleaning up the environment. Basil models actions either as arbitrary (constant or indeterminate, in the latter case input by the user at run-time), or as governed by events (such as a corner coming into contact with the guideline). Teacher begins by selecting the boxes; Basil classifies these as the input set. When Teacher leads Basil through the placement of the guideline's two end-points (Fig. 6c), Basil observes the absence of contact constraint, classifies the event as arbitrary and interrupts to ask (through a dialogue box, see Figure 8) whether the location is constant or a run-time input. Teacher indicates that both points are to be specified at run-time.

Teacher then leads Basil through the main iterative sequence (Fig. 6d-i). It is easy for Basil to observe that the objects he transforms are members of the input set, and that iteration terminates when every member has been processed. In general, however, selection and iteration depend on any number of properties of objects or situations. Thus iteration must be ordered and conditioned on events that Basil can sense by touch. A horizontal sweep-line serves this purpose, and also constrains the boxes' path of translation. Teacher makes Basil draw the sweep line near the bottom of the screen and indicates (through a dialogue box) that its initial placement is constant. Teacher then makes Basil grab the sweep-line at its mid-point handle and moves Basil and the line upwards until the latter touches the bottom edge of some box (Fig. 6d). This contact is the condition on which a box is selected for translation. When the line is swept past the last remaining box, Basil notes that the sweep-until-contact action will fail. This becomes his condition for terminating the loop.

Observe that this program applies the alignment constraint through goal-directed translation, where the goal is a visible contact. A suitable description of the goal, say "lower-left corner of box in grasp is coincident with some point on guideline", is invariant over iteration on the input set. Basil distinguishes this contact event and induces its invariance.

When the sweep-line touches the first box, Teacher has Basil grasp the box and move it rightward until its lower right corner touches the guideline, its bottom edge still on the sweep-line (Fig. 6e). Teacher then moves Basil back to grasp the sweep line and proceed to the next box (Fig. 6f). When Basil is made to select this second box, the action patently repeats that of selecting the first box. Consequently Basil conjectures a loop and predicts the translation that is to follow (Fig. 6g). The second box, however, must be moved to the left. Basil is biased towards easily generalizing directions of movement, so this does not faze him; he leaps out of Teacher's control and moves the box on his own. Teacher does not object; Basil has now learned the body of the loop and operates on the next box by himself (Fig. 6h-i).

After processing the third and final box, Basil recognizes that he cannot complete the action of moving the sweep-line as he has learned it; he terminates the loop on the condition

of being unable to perform its first step, and calls upon Teacher to demonstrate what to do. Teacher removes the sweep-line and guideline (Fig. 6j), and then announces that the lesson is over.

III REPRESENTING GRAPHICAL KNOW-HOW

Basil's problem domain comprises those composite graphic operations that can be accomplished by satisfying a sequence of positional constraints. People can communicate solutions to such problems to the computer by using the sort of visually explicit, procedural descriptions that characterize geometric methods of constructive proof. For example, to specify that two boxes are to be separated by a given distance, the instructor does not place them by eye, but instead uses a line of appropriate length as a spacer, placing one end at the first box and moving the second box to the other end. Instructors may use any drawing tools the computer provides, and are not restricted to the classical ruler and compass.

Insisting that only constructive methods be employed makes the transfer of graphical procedures from user to computer feasible. But will they be usable by the people they are meant to serve? — school-age children, office workers, draftsmen, and artists (not to mention computer scientists). Formal training may strongly influence the ingenuity and resourcefulness shown by people when conceiving geometric constructions. On the other hand, almost everyone shares the same phenomenology of geometry, described in [Freudenthal 67], which brings with it a set of powerful problem-solving methods. The basic constructs of what might be called "naive geometry" (after "naive physics" [Hayes 79]) are **order, measure** and **classification.**

Linear and cyclic **order** can be expressed constructively by using a sweep line [Preparata 85]. Objects are selected one after another, as they are touched by the line. Partial order in the plane corresponds to selection by an orthogonal pair of lines; the user must choose just how to arrange the two sweeps. With these devices an "untrained programmer" can specify such algorithms as finding the vertices of a convex hull (Jarvis' march), sorting boxes by height, or iterating some operation over a set.

Any ordering serves to distinguish certain points or thresholds, such as the mid- and end-points of a line segment, vertices of a polygon, or boundary between half-planes. Interactive drawing programs typically designate such points as "handles" whereby objects can be manipulated. Our system for inducing procedures pays attention to contact events which involve these handles. Many constraints can be expressed as a conjunction of items distinguished in this manner. Alignment, for example, is accomplished by translating points until they touch a boundary line. Constraints that do not involve handles (like dividing a line into thirds) require the teacher to invent appropriate constructions.

The notion of "order" dramatically highlights the chasm that separates human being from von Neumann computer. For the computer, naive sorting algorithms require $O(n^2)$ binary comparisons; clever ones, O(n log n). In the 2-dimensional graphic domain, human visual processing reduces these problems to O(n). Even more impressive is the ease with which a person can revamp his representation of the display space to accommodate different references of order. Suppose he wanted to sort objects with respect to an axis tilted at 45°; he need only cock his head (O(1)) and proceed as usual. The computer could take the same approach, in effect, by rotating the coordinate system (an O(n) operation), but first it must guess that this pre-processing is worth a try (O(immense!)), and then

determine (to the kth decimal place!) a suitable angle. The cost and unlimited variety of transformations required for a computer to mimic its human teacher make general function induction from input-output mappings utterly impractical. Explicit constructive methods are crucial to capturing functions which involve order.

What complicates the process of knowledge transfer is that nature has cunningly equipped human beings with built-in mechanisms for tasks which involve order relations. The eyes sweep across a set of objects or around a perimeter; visual acuity for detecting alignment is prodigious. In a study of users we found that people are least likely to express suitable constructions for tasks of this kind. They need considerable prodding before they are prepared to express the criteria used for ordering. Thus, our programming system presents the user with a metaphorical apprentice whose computational handicap is obvious — he is very near-sighted. This evokes the user's constructive ingenuity as an exercise in patience (and compassion?!).

One-dimensional **measures** include magnitude and angle. The former corresponds simply to the length of a line segment, the latter to the rotation of one line about another at a common end-point. With tools representing ruler and triangle, one can parameterize most of the constraints on a graphic design — such as the relative positions and sizes of the parts of a house.

Geometric equivalence, that is, **classification**, often depends upon a group of operations transforming one object into another [Freudenthal 67]. Everyday experience indicates that we tend to think in terms of constructing transforms. For example, we might verify that one triangle is similar to another by imagining them superimposed and then progressively scaled until their outlines merge. This epistemic compatibility bodes well for our programming system. Nonetheless, classification problems subsume those of order and measurement (eg. finding the vertices of a maximal hull). Moreover, between classifications that are governed by a simple construction (eg. points at equal distance from some reference) and those that are inexplicable (eg. nicely arranged points) are many problems that should be instructible but are very difficult to describe constructively (eg. groups of points that roughly balance each other in distribution).

Despite some deep problems, the use of explicit construction shows considerable promise as a means of transferring knowledge from human to computer. The psychological realism of procedures is also well supported. P. van Sommers has gathered considerable empirical evidence of the systematic, procedural nature of drawing as practised even by young children [van Sommers 84]. Hierarchical decomposition (nested iteration) is common practice. People tend to satisfy constraints in an orderly manner. Thus if a line is to branch off from another into empty space, it is almost always drawn from the stronger constraint (the contact with the parent line) to the weaker, even when this goes against the preferred stroke direction. People like to use visual contact events to delimit operations. A pyramid built of rectangles is typically drawn bottom up, so that the vertices of successive levels lie on those below. In experiments on the reconstruction of shapes from memory, van Sommers found that visual memory seems to record only scattered, local constraints. Nearly all subjects were able to draw some isolated joints of a triskele, for example, but almost none could reproduce the whole to a good approximation. Often, a subject would start out well but suddenly become completely confused.

The results of descriptive and quantitative studies support van Sommers' claims that drawing is primarily a process of accretion, and is highly optimized to minimize the number

and complexity of both mechanical and cognitive operations. Well-formed procedures can be created to model such processes providing they involve an appropriate progression of visuo-tactile constraints to be satisfied at each step. This corresponds to the constructive task model we propose. It leads naturally to a representation upon which a practical learning system can perform induction: namely, a sequence of drawing strokes or affine transformations delimited by contact events or by explicit justification (conscious decision). We should note, however, that van Sommers studied drawing on paper. The ergonomics when drawing with a mouse must be quite different from those with a pencil. Moreover, the ability to move what one has just drawn affects the application of end-control constraints. We assume that while certain ergonomic considerations are somewhat relaxed in computer drawing, the accretive, constructive model still applies. This will be investigated empirically in the near future.

IV META-MOUSE: BODY AND BRAIN

Whereas computer drawing environments provide a rich selection of primitive objects and operations, our experimental domain accommodates only two object types, line and box, and two canonical transforms, translation and scaling. An additional composite transform, called "reshape", can effectively scale and rotate a line by moving one end-point. These are the basis of the drawing world, a program called A.Sq, whose user interface methods resemble those of MacDraw [Cutter 87]. The user of A.Sq can create objects by rubber-banding, select one or more of them with a gravity pick, and transform selected objects by moving their handles. End-user programs for this system are composed of these same operations.

Basil, the Meta-Mouse for programming in A.Sq, mimics the behavior of the standard mouse pick/locator device. Basil's body can either be a Turtle's (Figure 4), or a simplified icon that shows just position and orientation, with controls on the keyboard. The basic operators are to move forward, back, left, right, or along an unconstrained path, and to grasp or let go. In order to transform an object, Basil grasps one of its handles and carries it to its destination (see Figure 5). Note that Basil, like his teacher, can only grasp an object by its handles. To facilitate programming problems that involve rotational order (eg. convex hull), Basil can be instructed to perform true rotation. To rotate an object, he grasps its translation handle and turns himself.

The actions of a program have determinate parameters, some of which must be variable for the program to be general. Moreover, it must be possible to determine which action to perform when branching occurs. Basil therefore models actions as (pre-condition, operation, path, post-condition) tuples, in the manner of STRIPS [Fikes 71]. Operations are Meta-Mouse commands; paths are directions of motion; and conditions are generalizations of Basil's sensory feedback (see Figure 7). For example, a program step Basil has learned might check that he is facing right and grasping a box by its translation handle, and if so, direct him to move forward until the bottom-right handle of the thing he is grasping (the box) touches the left side of any other box. The numerical parameters of the corresponding A.Sq translate command are calculated from the distance a ray travels from Basil's snout along the specified path until intersecting a "left edge".

Basil executes (predicts) a program step provided that 1) its pre-conditions are a generalization of the current sensory state, and 2) its post-conditions are attainable.

Otherwise Basil checks any alternative steps; if none remain, he asks the user to show him what to do in this novel situation and creates a new branch on the current condition, or, if the operation turns out to be one of the rejected alternatives, disjoins the current situation with the pre-conditions already learned.

A program learned by Basil is represented as a non-deterministic automaton that grows as needed. The automaton is a directed cyclic graph: conditional branches and their return to parent sequences are lattice-like structures; loops are cycles within a lattice. This permits a simple, reliable algorithm to construct programs from traces. A novel action, performed by the Teacher, is simply appended to the current branch. Basil conjectures a linkage of branches (forms a lattice) if two "sufficiently similar" events occur. This means that if Teacher adds a step to the end of a branch and Basil finds a step with matching pre- and post-conditions, Basil conjectures a link to this step, and predicts its successor. If Teacher accepts the prediction, Basil takes the conjecture as correct. Our prototype system never retracts a linkage once it has been accepted in this way: if Teacher rejects subsequent predictions, Basil attempts to construct a conditional branch around the rejected actions. In keeping with van Sommers' model of the hierarchical decomposition of drawing tasks [van Sommers 84], we bias Basil's search for linkages towards actions that are "near," meaning that they have more predecessor actions in common. He looks first within the current branch (for a loop), then sibling branches, parent branches, and so on.

Basil's sensory feedback comprises current position, heading and tactile state. In keeping with the phenomenological model, the last is by far the most important. Position is used only as a constant, when Teacher specifies "same-place" generalization. Heading is used only when Basil has been rotating and must return to some previous orientation. Basil's tactile state is governed by what he is grasping (he may grasp only one object) and touching at his snout (several objects may be touched). Tactile information comprises 1) the specific part of each object touched (or grasped); 2) the properties of each object touched (or grasped), including their type and the locations of their handles; and 3) spatial relations, expressed in tactile terms, between an object grasped and any other objects it touches (see Figures 5 and 7). Note carefully the limits placed on Basil's awareness of touch: he senses only at his snout (current exact position), and is not aware of contacts between other objects and what he is touching, nor of contacts between other objects and those that touch what he is grasping. These extra data are not needed to model successive constraints, provided that Basil can always check for significant contacts by being moved to the location where they might occur. This simplification is intended to prevent the teacher from believing that Basil could analyze the entire display. Teacher is told that Basil only pays attention to things he bumps into.

Basil's working memory has four conceptual divisions. Long-term memory contains the procedure as learned so far; from this Basil conjectures block structure linkage and predicts actions. The medium-term state records what objects have been created during the current trace; thus Basil recognizes a constructor, such as a sweep-line, when he encounters it, and can be taught to return to that particular constructor. Short-term memory records the identities of objects most recently touched or grasped; this permits letting go of something in order to investigate it further by "sniffing" around its perimeter. The immediate memory contains the current sensory feedback.

Any conflict between tactile feedback and position, heading, or path, is resolved by generalizing the last three. Generalization of tactile conditions (touch relations) is not

undertaken lightly. While generalization by inference (ie. climbing a hierarchy) can greatly accelerate learning, the specter of over-generalization and the difficulty of correcting it remain. Moreover, daring, successful generalizations may delude the teacher regarding Basil's intelligence, while failures may result in a loss of confidence. Our prototype system, therefore, does minimal generalization on touch relations, merely disjoining relations that have occurred. In future we will experiment with other generalization heuristics.

V MEETING THE FELICITY CONDITIONS

The user of a system for programming by example has much weaker assumptions about the system than a programmer working in a formal language. Its limited powers of induction are revealed gradually, and typically in disastrous occurrences that the user finds hard to interpret. It has been argued that the relationship between programmer and system is best understood as that of teacher and apprentice, and hence that the perspicuity of the teaching metaphor is vital to the success of the system [MacDonald 87].

We studied the performance of graphical tasks by a diverse group of people using MacDraw (no Meta-Mouse). Each subject was given one hour to complete seven tasks; activity was recorded using a commercial system for programming by example [Tempo 86]. The results of this informal study indicate that Meta-Mouse must clearly inform the teacher of his awareness, abilities, and limitations as a pupil. Meta-Mouse promotes the illusion that he is paying attention to the teacher's activities by tracking the movement of the physical drawing tool (mouse or stylus). He also transmits feedback from the learning system. When he registers contact with an object (bumping his head) he signals by flashing; when an action has no visible end-control, he requests an explanation via a simple dialog box.

Any teaching situation has a complex communication protocol that can be modelled as a set of "felicity conditions" whose primary purpose is to constrain the possible interpretations that a pupil must consider in assimilating each component of a lesson [Van Lehn 81]. Variants of four felicity conditions that Van Lehn found important in teaching arithmetic to children are operationalized in our learning system and its user interface. Although capable of adopting these felicity conditions, which are deeply ingrained in our social culture, our users, who are not professional teachers, will surely have difficulty satisfying them. On the other hand, it is self-defeating to impose artificial rules, and impractical to build a learning system that can accommodate greater variability. (Such a system would have to be able to induce general functions in a noisy environment, a technological feat far in advance of the state of the art [eg see Andreae 85, Langley 87].) To resolve the dilemma, we place the burden of checking that instruction is satisfactory upon Meta-Mouse, which is at least capable of knowing when he cannot interpret the teacher's actions. The result is an intensely interactive system.

The fundamental felicity condition is that lesson materials be **correct**. Since correctness is relative to the teacher's intentions, this must remain his responsibility. Meta-Mouse is, however, able to recognize contradictions. If, in a given situation, he predicts an action that differs from Teacher's, and cannot discriminate the model situation from the current one, then Teacher is informed that he is either inconsistent or has failed to give

enough information to discriminate. The system also filters out minor positioning errors by using a "gravity" function.

The **show work** condition is the very basis of our approach. The system constructs a procedure to satisfy a sequence of constraints; this procedure integrates one or more action traces with control structures for branching and looping. It does not accept "before" and "after" pictures; but instead must be shown how the transformation is done. The system acknowledges actions performed while Meta-Mouse is attached to the teacher's input device, to promote the illusion of learning by doing. The subjects of our user study were not told to teach tasks; indeed, they were not very familiar with them. Nonetheless, after some trial and error, they developed procedures that incrementally transformed inputs to outputs, rather than simply draw the outputs from scratch. However, it cannot be guaranteed that all transformations will be constructed. For example, the teacher might "stretch a line to fit between two boxes" by drawing a new line that fits; the system would observe no relationship between the two lines.

In order to induce constants, inputs and variables without resorting to sophisticated function induction, the **no invisible objects** rule is enforced (this contributes to the "show work" maxim). Invisible objects are the constants and variables of relations, such as distance, that are not explicitly mentioned in a description. Eliminating invisible objects entails showing work at a finer granularity. Our user study indicated that, while some more experienced users employ explicit constructions, most are inclined to measure gaps and alignment by eye. The learning system, through Meta-Mouse, encourages explicitness. Thus, as a trace is carried out, the system ensures that it gets all required parameters. The teacher knows that Basil is near-sighted and must "feel out" relations. Variables are (the relevant attributes of) constructed objects whose end-positions are determined by contact with other objects. Seemingly arbitrary creations — drawing operations without tactile end-control — are interpreted as either constants or inputs; the Meta-Mouse asks which (see Figure 8). Teacher must take care to avoid coincidental contacts when siting the start or end-point of a line, but this restriction is obvious and easily satisfied.

The final felicity condition is **minimal activity**. (This is a variant of Van Lehn's "minimal disjunct" appropriate to a system with perfect memory.) Orderliness and consistency in the actions presented enable Meta-Mouse to identify conditional branches and loops. Extraneous activity not only adds useless complexity to the program but may subvert sequence matches and prevent control structures from being induced. The subjects of our user study engaged in a great deal of extraneous activity while sorting out their constructive methods. We classified this as: 1) missteps (actions quickly retracted); 2) experiments (action sequences retracted, often performed on duplicates of input data); 3) bustle (useless actions due to confusion or to pass time while thinking); and 4) superseded (user has revised method without retracting actions). Although the user is likely to plan more carefully when playing the role of teacher, we should not expect him to be so very orderly.

Basil attempts to keep instruction focussed by predicting actions whenever possible. As the teacher works through the first example trace, the system matches the current action and its pre-conditions with those that have come before. If he can predict iteration, Basil detaches himself from teacher control and carries out predicted actions until he meets an unrecognized situation or the teacher rejects his prediction by grabbing his tail. Similarly,

Basil executes his previously learned procedure on a new set of data up to the point where a novel situation (or disagreement with the teacher) indicates the need for a conditional branch. This practice suits the teacher-apprentice metaphor. If the teacher wishes to do something extraneous, such as an experiment, he can put the Meta-Mouse into an idle state for the duration.

VI CONCLUSIONS

The design of Basil has been strongly influenced by the results of our experimental study of user behavior in constructive graphical tasks. We have not completed a proper evaluation of how people interact with the new drawing tool. Nevertheless, we can already draw some conclusions from the work.

First, user interaction can augment or replace domain knowledge in constraining the massive searches incurred by function induction without requiring that the user manipulate or even understand the internal representation. The three-way trade-off between search space, ease of teaching, and built-in knowledge can be readily investigated in the graphical domain. Second, an eager learner can reduce teacher noise and enforce felicity conditions. The learner's actions must be clearly visible, and it is of course vital that a convenient "undo" facility be provided to control its impetuosity.

Finally, much knowledge acquisition research aims to improve the bandwidth of knowledge extraction by more closely modelling the expert's knowledge ordering principles, providing him with more specialized tools for expressing himself. These tools are nonetheless formalizations of knowledge into which the expert must translate what he knows. Our approach is intended to leave the expert in his domain, working as a he usually does except for the intrusion of an apprentice and its occasional questions and mistakes. This metaphorical teaching process emulates a long established practice among human beings for transferring expertise and should prove more effective than formalized dialogues in a number of domains.

ACKNOWLEDGEMENTS

This research is supported by the Natural Sciences and Engineering Research Council of Canada. We gratefully acknowledge the key role Bruce MacDonald has played in helping us develop these ideas, and the stimulating research environment provided by the Calgary Machine Learning Group.

REFERENCES

[Abbott 1884] E. A. Abbott. *Flatland — A Romance of Many Dimensions.* Signet Classics edition. New York. 1984.

[Abelson 80] H. Abelson, A. diSessa. *Turtle Geometry.* MIT Press. Cambridge MA. 1980.

[Andreae 85] P. M. Andreae. "Justified Generalization: Acquiring Procedures from Examples." PhD dissertation. Department of Electrical Engineering and Computer Science, MIT. January 1985.

[Attardi 82] G. Attardi, M. Simi. "Extending the power of programming by examples." *ACM Conference on Office Information Systems,* pp. 52-66. June, 1982.

[Barber 83] G. R. Barber. "Supporting organizational problem solving with a workstation." *ACM Trans Office Information Systems, 1 (1)*, pp. 45-67. January, 1983.

[Bier 86] E. A. Bier. "Snap-dragging." *Computer Graphics: Proc. ACM SIGGRAPH.* Dallas. August 1986.

[Borning 86] A. Borning. "Defining constraints graphically." *Human Factors in Computing Systems: Proc. ACM SIGCHI '86.* Boston. April 1986.

[Croft 84] W. B. Croft, L. S. Lefkowitz. "Task support in an office system." *ACM Trans Office Information Systems, 2 (3)*, pp. 197-212. July, 1984.

[Cutter 87] M. Cutter, B. Halpern, J. Spiegel. MacDraw. Apple Computer Inc. 1985, 1987

[Fikes 71] R. E. Fikes and N. J. Nilsson. "STRIPS — a new approach to the application of theorem proving to problem solving." *Artificial Intelligence, vol. 2*, pp. 189-288. 1971.

[Freudenthal 67] "Geometry - a phenomenological discussion." in *Fundamentals of Mathematics, vol. II.* ed. H. Behnke, F. Bachmann et al, transl. S. H. Gould. MIT Press. Cambridge MA. 1974.

[Fuller 86] N. Fuller, P. Prusinkiewicz. "L.E.G.O.—an interactive graphics system for teaching geometry and computer graphics." *Proc. CIPS Edmonton.* 1986.

[Halbert 84] D. C. Halbert. "Programming by Example." Research Report OSD-T8402. Xerox PARC. Palo Alto CA. December 1984.

[Hammer 80] M. Hammer, J. S. Kunin. "Design principles of an office specification language." *Proc National Computer Conference, 49*, pp. 541-548. 1980.

[Hayes 79] P. J. Hayes. "The naive physics manifesto." *Expert Systems in the Microelectronic Age*, pp. 242-270. ed. D. Michie. Edinburgh University Press. Edinburgh. 1979.

[Kunin 82] J. S. Kunin. "Analysis and specification of office procedures." PhD Thesis. Department of Electrical Engineering and Computer Science, MIT. February 1982.

[MacDonald 87] B. A. MacDonald, I. H. Witten. "Programming computer controlled systems by non-experts." *Proc. IEEE SMC Annual Conference.* Alexandria VA. October 1987.

[Myers 86] B. A. Myers, W. Buxton. "Creating highly-interactive and graphical user interfaces by demonstration." *Computer Graphics: Proc. ACM SIGGRAPH '86.* August 1986.

[Nix 84] R. Nix. "Editing by example." *Proc 11th ACM Symposium on Principles of Programming Languages*, pp. 186-195. Salt Lake City UT. January 1984.

[Papert 80] S. Papert. *Mindstorms.* Basic Books. New York. 1980.

[Preparata 85] F. P. Preparata, M. I. Shamos. *Computational Geometry.* Springer-Verlag. New York. 1985.

[Smith 75] D. C. Smith, "Pygmalion: a Creative Programming Environment." Report No. STAN-CS-75-499. Stanford U. 1975.

[Suchman 87] L. A. Suchman. *Plans and Situated Actions — the Problem of Human-Machine Communication.* Cambridge University Press. 1987.

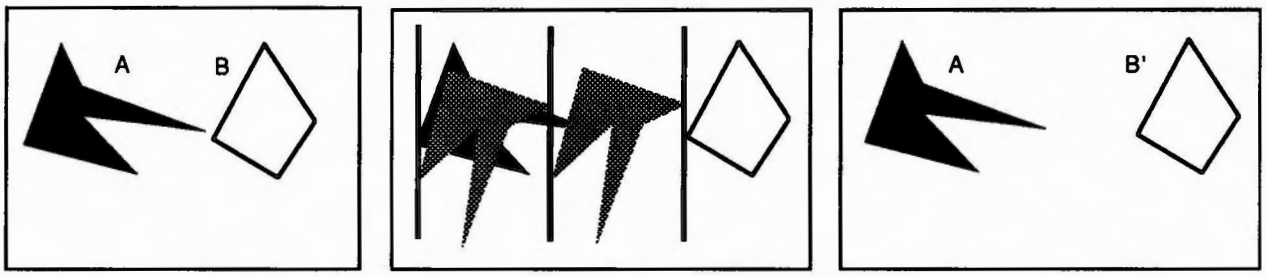
[Tempo 86] Tempo. Affinity MicroSystems Ltd. Boulder CO. 1985, 1986.

[Van Lehn 83] K. Van Lehn. "Felicity Conditions for Human Skill Acquisition: Validating an AI-Based Theory." Research Report CIS-21. Xerox PARC. Palo Alto CA. November 1983.

[van Sommers 84] P. van Sommers. *Drawing and Cognition.* Cambridge Univ. Press. Cambridge UK. 1984.

[Witten 87] I. H. Witten, B. A. MacDonald, S. Greenberg. "Specifying procedures to office systems." *Proc International Conference on Computer-based Tools for Information Systems Analysis, Design and Implementation.* Loughborough UK. April 1987.

[Zisman 77] M. M. Zisman. "Representation, Specification, and Automation of Office Procedures." PhD Dissertation. Wharton School, U of Pennsylvania. 1977.

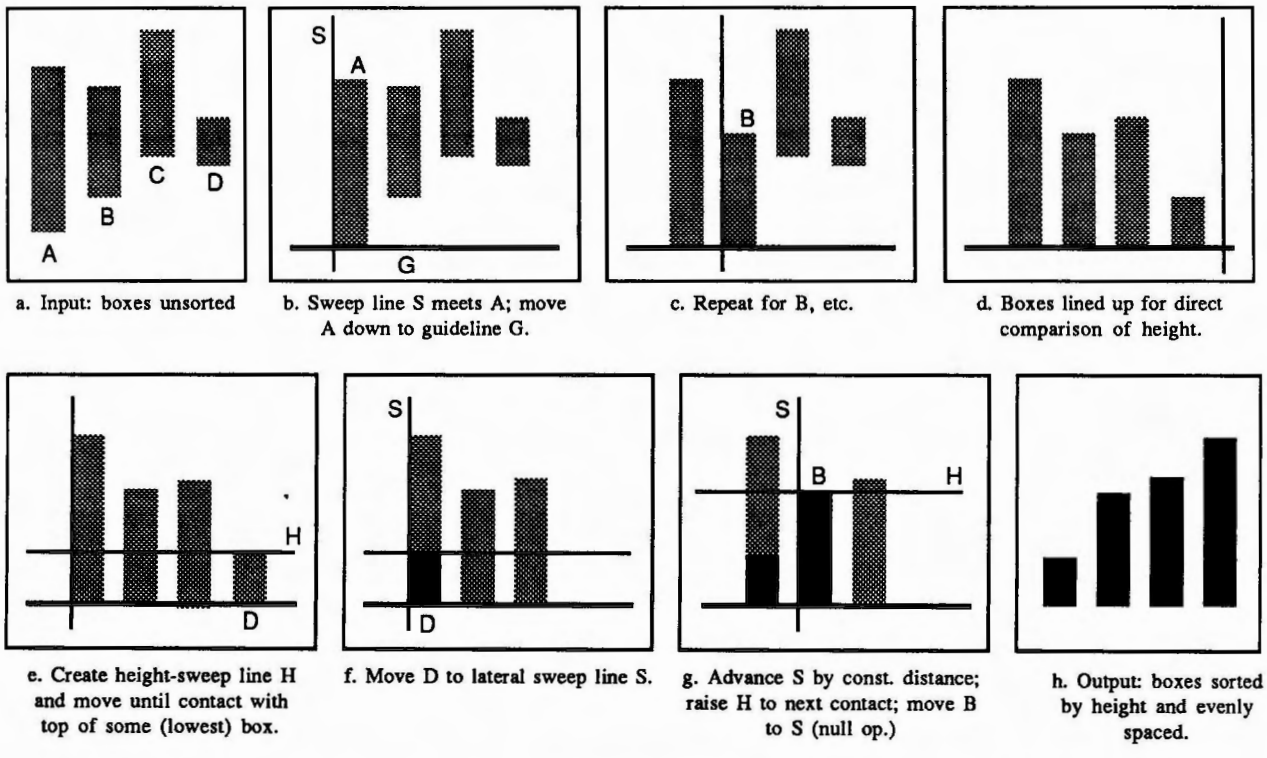


a. Input: two polygons

b. Construction: measure height of A by duplicating and rotating 90°; translate B.

c. Output: A and B' (=B transformed)

Figure 1. Graphical task: translate polygon B such that distance from left extreme of A to left extreme of B is 2 x height of A.



a. Input: boxes unsorted

b. Sweep line S meets A; move A down to guideline G.

c. Repeat for B, etc.

d. Boxes lined up for direct comparison of height.

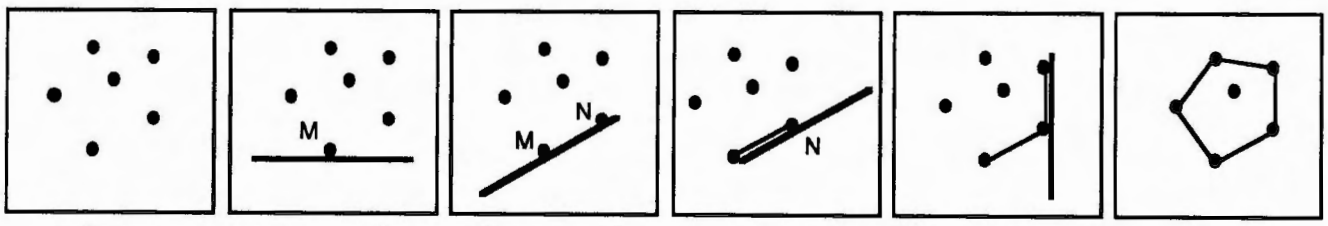
e. Create height-sweep line H and move until contact with top of some (lowest) box.

f. Move D to lateral sweep line S.

g. Advance S by const. distance; raise H to next contact; move B to S (null op.)

h. Output: boxes sorted by height and evenly spaced.

Figure 2. Task: given a set of boxes, sort them in order of increasing height.



a. Input set

b. Bring sweep-line up to lowest member, M

c. Rotate sweep line about M to contact N

d. Connect M, N; center sweep at N

e. Repeat c, d from N

f. output

Figure 3. Task: given a set of points, find their convex hull.

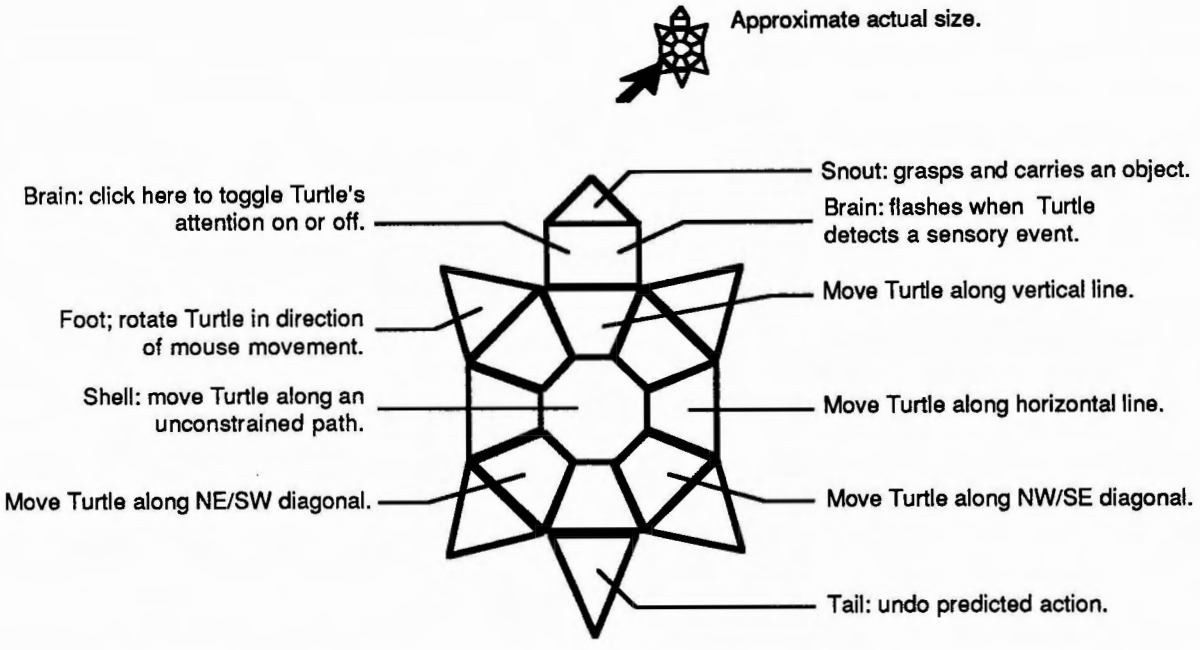
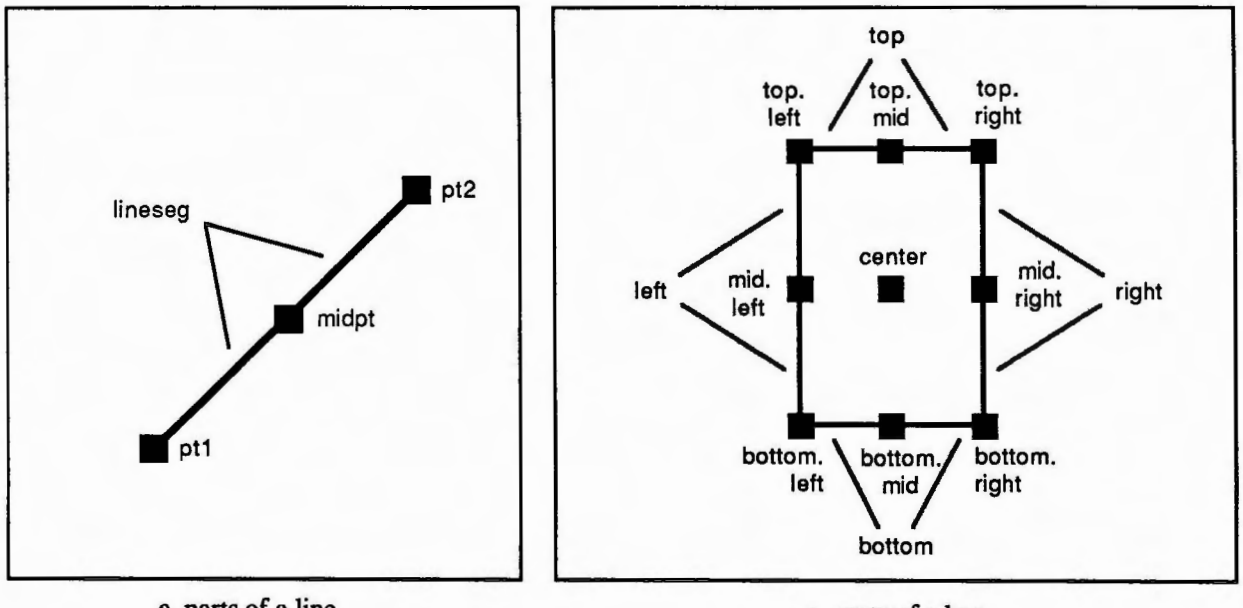


Figure 4. Anatomy of Meta-Mouse, a moveable menu that sometimes runs away.



a. parts of a line

a. parts of a box

Figure 5. Parts of objects.

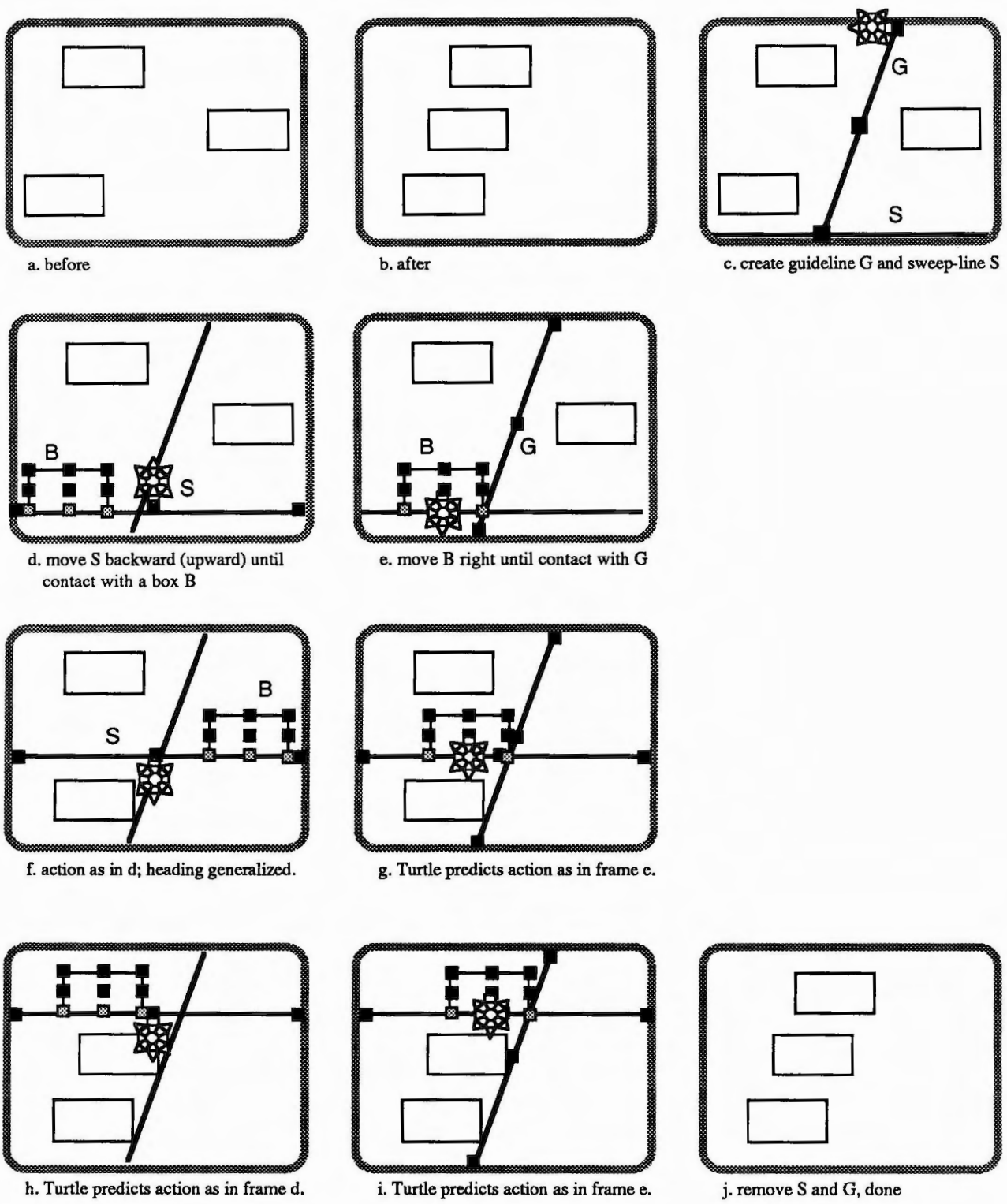
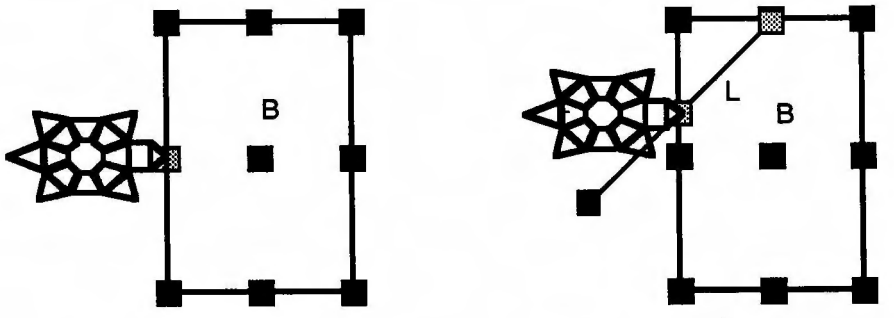


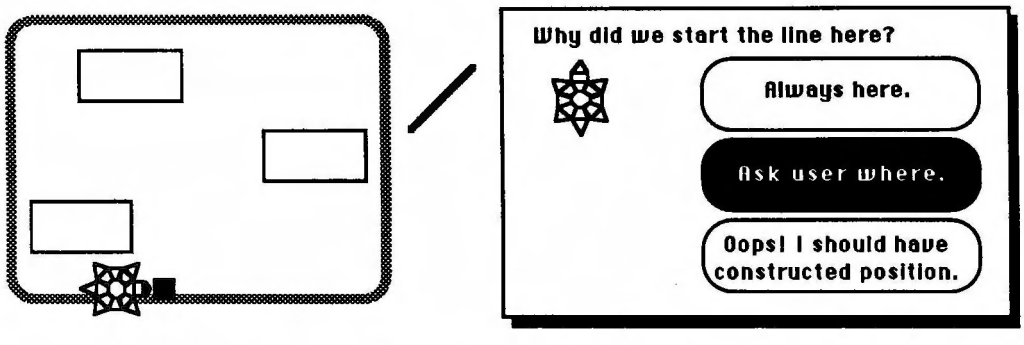
Figure 6. Action / teaching trace of box-to-line procedure.



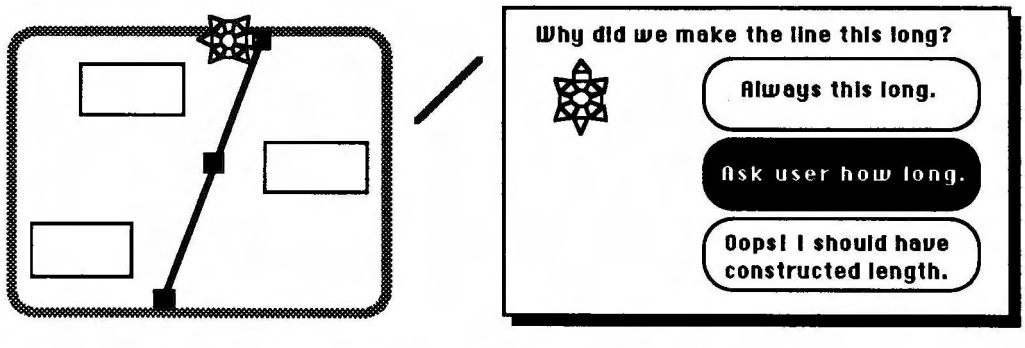
touch (self, (B = a-box).mid.left)

grasp ((L = a-line).midpt)
touch (L.farpt, (B = a-box).top.mid)
touch (L.midpt, B.?.left)

Figure 7. Description of contact events for pre- and post-conditions.



a. query for explanation of line origin



b. query for explanation of line termination

Figure 8. Asking Teacher to explain seemingly arbitrary decisions.