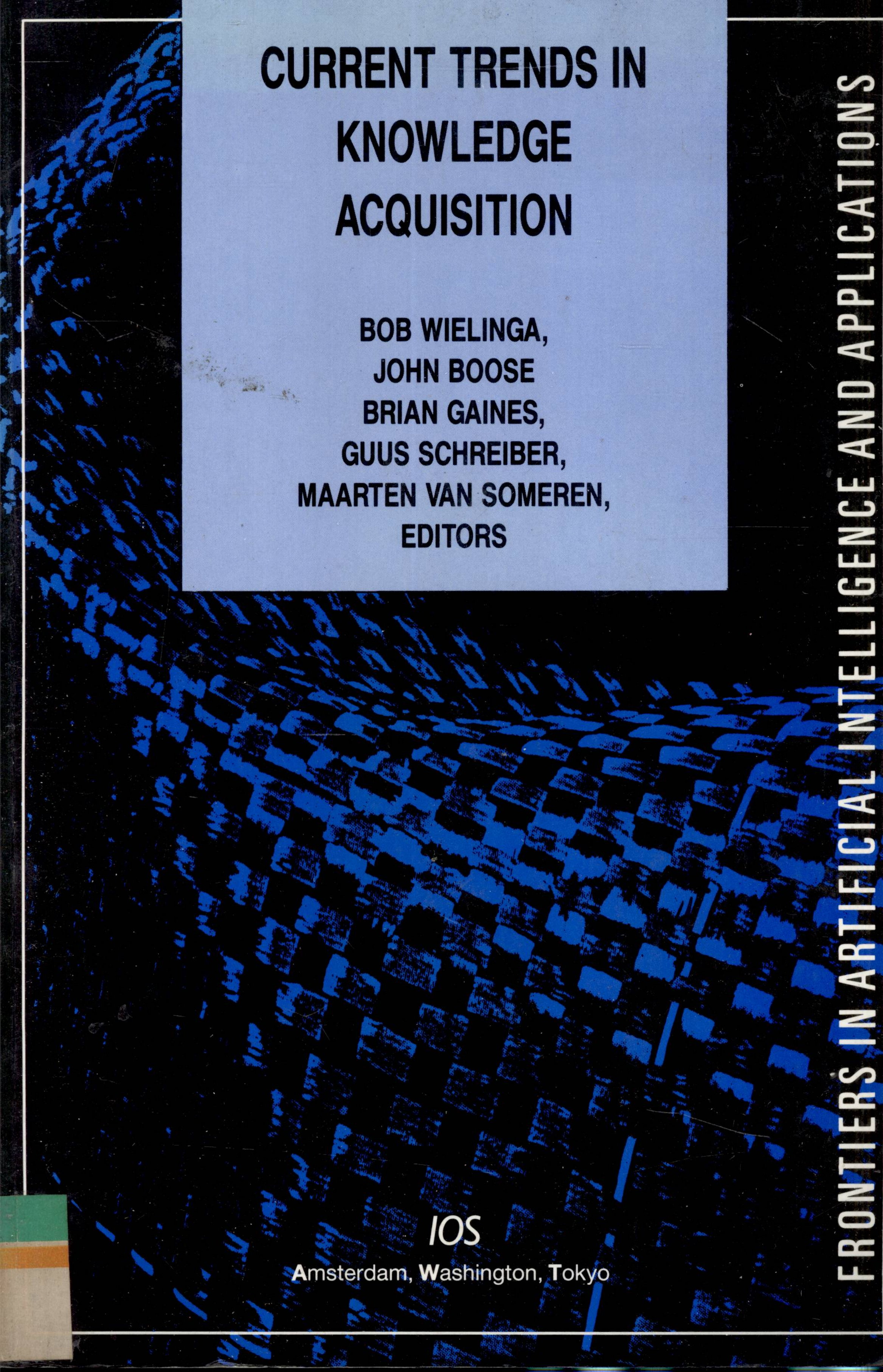
CURRENT TRENDS IN KNOWLEDGE ACQUISITION

BOB WIELINGA,
JOHN BOOSE
BRIAN GAINES,
GUUS SCHREIBER,
MAARTEN VAN SOMEREN,
EDITORS

IOS
Amsterdam, Washington, Tokyo

FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS



TP18
W646

Current Trends in Knowledge Acquisition

Edited by

Bob Wielinga
University of Amsterdam

John Boose
Boeing Computer Services, Seattle

Brain Gaines
University of Calgary

Guus Schreiber
University of Amsterdam

Maarten van Someren
University of Amsterdam

IOS Press

1990

Amsterdam • Washington, DC • Tokyo

ISBN 90 5199 036

Publisher:

IOS
Van Diemenstraat 94
1013 CN Amsterdam
Netherlands

Distributor in the U.S.A. and Canada:

IOS Press, Inc.
P.O. Box 10558
Burke, VA 22009-0558
U.S.A.

Distributor in Japan:

IOS Japan Dept.
Highway Development Co. Ltd.
1st Golden Bldg, 8-2-9 Ginza
104 Tokyo - Chuoku
Japan

PRINTED IN THE NETHERLANDS

CURRENT TRENDS IN KNOWLEDGE ACQUISITION

FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS

ISSN: 0922-6389

Preface

Knowledge acquisition has become a major area of artificial intelligence and cognitive science research over the past decade. The growing number of publications and workshops is a clear sign of this development. This book contains the contributions to the fourth European Knowledge Acquisition Workshop (EKAW-90) held in the Netherlands. The papers form the basis for discussions on issues in knowledge acquisition, and as such they reflect both the current state of the field and the opinions of the authors on important research topics. The papers in this book show that the area of knowledge acquisition for knowledge-based systems is still a diverse field in which a large number of research topics are being addressed. However, several main themes run through the papers.

Knowledge integration First, the issue of integrating knowledge from different sources and K.A. tools is a salient topic in many papers. Brazdil and Torgo explicitly address the problem and propose a method based on the use of empirical evidence as a criterion for integration. Shadbolt and Wielinga describe a scenario in which integration is presented as a knowledge-based activity. Another paper from the ACKnowledge project by Eggen, Lundteigen and Mehus, discusses integration from a computational viewpoint. Van Someren, Zheng and Post discuss how three types of knowledge obtained through different knowledge acquisition techniques can be combined in an integrated system.

Several papers deal with advanced workbenches in which different tools for knowledge acquisition are integrated. Gaines and Linster describe the integration of the tools from KSS0, based on the repertory grid approach, with the representation system Babylon through hypertext. They advocate an integration methodology in which existing tools are integrated within a common environment. The Keats system also combines several knowledge engineering tools in one environment although emphasis of Motta's paper is on methodology. The Shelley system (Anjewierden et.al.) achieves a large degree of integration of K.A. tools through the use of a repository of common persistent objects representing the results of various K.A. tools.

Knowledge modelling A second major topic in the papers is that of *knowledge modelling.* Recent research in knowledge-based systems emphasises the use of generic models of reasoning and its underlying knowledge. Such models are generally placed at the knowledge level. In spite of the growing consensus on the importance of knowledge models in knowledge acquisition, little has been done to create a unified framework for knowledge modelling. Karbach, Linster and Voss present a comparison of four influential approaches to knowledge modelling and its role in knowledge acquisition. Even if one disagrees with some of their conclusions, the paper is an important contribution to the discussion of the nature of such models used in knowledge acquisition.

An important trend in the area of knowledge modelling aims at the formalisation of knowledge models. Wetter presents a first order logic representation of KADS models and compares it with earlier attempts to formalise KADS models by Wielinga et. al. Kowalczyk and Treur use methods of logic to construct formal specifications of generic tasks. Balder and Akkermans describe a tool which can support the structuring of formal specifications of KADS knowledge models.

Techniques and tools for knowledge elicitation Where the field of knowledge acquisition was without tools and techniques a decade ago, now there is a rapidly growing body of techniques and tools. Apart from the integrated workbenches already mentioned above, several papers in this book present new tools. Major and Reichelt describe a tool that supports knowledge elicitation through laddering. Boose, Shema and Bradshaw discuss the use of a variety of knowledge acquisition tools in order to capture design knowledge in a *corporate memory facility.*

Meyer, Booker and Bradshaw address the problem of *bias* in knowledge elicitation. They describe a six-step method for handling bias in a particular application. Hoppe approaches the important issue of validation of knowledge as a problem of acquiring the user intention and establishing whether the user intention meets the actual system. Davis and Bonnell present a knowledge acquisition method *-Conceptual Abstraction-* derived from semantic modelling for databases.

Two papers deal withy methods for acquiring knowledge from text. Schmidt and Schmalhofer describe a systematic method for knowledge acquisition from text in which the domain expert plays an important role. Ruqian and Cungen approach the problem of the acquisition of domain knowledge from books through the definition of an intermediate language which is close to the original natural language, but which can be automatically translated to a knowledge base. The translation from the original text to the intermediate language is done by the knowledge engineer.

Knowledge acquisition and machine learning Although knowledge acquisition and machine learning have been considered as separate subfields of AI, there is a recent tendency for the two fields to come together. The ACKnowledge project represented by the papers of Eggen et. al. and Shadbolt and Wielinga, combines machine learning techniques with more conventional knowledge elicitation techniques. Aamodt presents a framework in which reasoning, problem solving and learning together form a knowledge intensive system that can acquire knowledge from its own experience.

Induction has been shown in the past to be an effective means of knowledge acquisition by itself. Applying induction techniques usually requires considerable engineering of the data relies on a substantial amount of data. A major problem of applying induction techniques to knowledge acquisition problems is that the results of induction are often difficult to understand for expert, the end user of the knowledge base and also for the knowledge engineer. This makes it difficult to integrate induction with manual knowledge engineering and causes problems with explanation for users. Van Someren et.al. describe these problems and propose a solution by decomposing the problem into subproblems that can adequately de dealt with by either induction or manual techniques. Kononenko shows that experts prefer naive bayesian statistics over decision trees and Nunez shows how extra knowledge such as costs, hierarchies of descriptors, etc. can be used to obtain more comprehensible results of induction.

We are sure that the reader will find in the material presented in this book stimulating new ideas and topics for further research in this challenging field.

Acknowledgement

As editors we would like to thank the following members of the EKAW-90 programme committee for their assistance in reviewing and selecting the papers for this book.

Tom Addis, University of Reading
Guy Boy, Centre d'Etudes et de Rech. de Toulouse
Jeffrey Bradshaw, Boeing Computer Services
Jean Gabriel Ganascia, Université Paris-Sud
Yves Kodratoff, Université Paris-Sud
Marc Linster, GMD, St Augustin
John McDermott, Digital Equipment Corporation
Ryszard Michalski, George Mason University
Katharina Morik, GMD, St Augustin
Nigel Shadbolt, University of Nottingham
Mildred Shaw, University of Calgary

The editors
May 1990

Contents

A Computational Model of Knowledge-Intensive Learning and Problem Solving

Agnar Aamodt

Knowledge Engineering Laboratory, ELAB-RUNIT, SINTEF
N-7034 Trondheim-NTH, Norway
and
Department of Electrical Engineering and Computer Science
University of Trondheim
agnar.aamodt@elab-runit.sintef.no

Abstract. If knowledge-based systems are to become more competent and robust in solving real world problems, they need to be able to adapt to an evolving domain and a changing environment. This paper proposes a computational model - a framework - for knowledge-intensive problem solving and learning from experience. The model has been instantiated in an architecture for knowledge-intensive case-based reasoning and learning called CREEK (Case-based Reasoning through Extensive Expert Knowledge). The importance of a thorough, extensive knowledge model to support the reasoning and learning processes is emphasized. In case-based reasoning a problem is solved by retrieving a similar past problem case, and using this case in solving the new problem. Learning becomes a process of extracting relevant information from a problem just solved, and integrating the new case into the existing case-base. The computational model presented combines case-based learning and reasoning with model-based and rule-based methods. The type of problems addressed are problems within open, weak theory domains.

Keywords: *Machine Learning, Knowledge modelling, Case-Based Reasoning.*

1. Introduction

1.1. General

Knowledge acquisition for high quality, expert level, knowledge-based systems is a difficult and time consuming effort. Such systems are aimed at solving complex real world problems - like medical diagnosis, fault-finding and repair in industrial processes, design of technical production equipment, planning of financial investments. They may have to cope with problems on the border of (or slightly outside) their special subject domain. The systems will have to operate in a continually evolving environment, which requires frequent updating and refinement of a system's knowledge and problem solving ability. Ideally, a system should be able to update its knowledge after each problem solving session. The problem of *knowledge maintenance* is therefore a major challenge for the AI community. If a system is to continually maintain and improve its problem solving competence, methods for *automated learning from experience* is required. The ability to learn from each attempt to solve a problem during normal run-time operation will be referred to as *sustained learning*. We argue that if such a learning process shall be effective and useful, it should be based on - and focused by - a thorough understanding of the subject domain. That is, an underlying deep knowledge model should guide the learning of operational knowledge, as well as the problem solving that makes use of it.

People are good problem solvers because we base our methods on a general understanding of the world we operate in. We also maintain a memory of past problem-solving episodes integrated into this fundamental knowledge structure. While solving problems, we are frequently reminded of similar past problems. Using more general knowledge as support, we are able to adapt the solution (or solution path) of the retrieved case to solving the new problem. Problem solving strategies help us to focus the use of knowledge within the current context and according to our goals. Through success and failure in achieving our tasks, we learn to do things better next time.

This paper describes a computational framework for knowledge-intensive learning and problem solving. The model combines *case-based reasoning* [28, 27, 11a, 12a] with methods that utilize generalized knowledge, i.e. deep models and heuristic rules. A system architecture, called CREEK, has been developed, based on the framework[1]. A comparison of the architecture with two other relevant approaches to knowledge-intensive case-based reasoning - the PROTOS [4, 5] and CASEY [14] systems - is made in [3], and a general overview of the approach with an example from modelling of a mud diagnosis system is given in [2].

1.2. Problem Solving in the Real World

Most real-world problems that knowledge-based systems are addressing, are *open problems*. An open problem is characterized by having a weak or intractable domain theory. That is, there is no theory that completely describes the problem domain, and from which correct solutions can be derived or proved. Typical weak theory domains are medical diagnosis, geological interpretation, investment planning, and most engineering domains (i.e. domains that involve interaction with the external world). A weak domain theory is characterized by uncertain relationships between its concepts. The stronger the theory, the more certain are its relationships. At the very far end of this spectrum are the complete theories. The relationships of complete theories are certain, e.g. as expressed by standard logical or structural relations. Domains with strong domain theories are, for example, mathematical domains, 'block worlds' and some games - like checkers and chess. Even some strong theory domains may incorporate problems that turn out to be open when addressed by a problem solver. Chess, for example, has a strongest possible - a complete - domain theory, but solving the problem of chess-playing by an implementation of the theory is violated by its intractability. The concept of a 'winning plan' in chess is theoretically deducible, but there is no efficient algorithm to infer it in the general case.

Adding to the problems of weak or intractable theories, problem solving in the real world is, in general, complicated by incomplete and noisy problem specifications. This puts two important requirements on an intelligent problem solver:

- The ability to infer missing information, to generate and check expectations, and to justify whether a piece of information is relevant or not.
- The ability to interact with the information source (usually a human being) in order to extract additional information, and for confirmation or rejection of dubious conclusions.

Hence, a real world problem solver should be able to dynamically change the scope and goals of a problem during problem solving, and to deal with the contradictions and inconsistencies that are introduced in this process. The basic method that enables a system to infer meaningful results, justify its own decisions, and interact intelligibly with the user, may be viewed as a process of *hypothesis generation* followed by *producing and evaluating explanations*. In real world, weak theory domains, it is seldom the case that an inferred result can be *proved* true or false. Instead, the problem solver has to produce and justify its conclusion by explaining to itself why the conclusion is a plausible one (see, e.g., [32]). A result derived from a previous problem solving case, for example, may need to be supported by an explanation based on general domain knowledge. The emphasis on explanation in the reasoning process requires a thorough and deep knowledge model in order to produce plausible, meaningful explanations.

1.3. Sustained Learning

A goal of machine learning research is to enable self-learning in computer systems. This goal is approached by developing theories and methods for processing and retaining observations (facts that are input to the system) in a way that enables them to be used in future problem solving. This is done either by inducing and saving generalised forms of the observed facts, or by keeping them as concrete cases from which solutions to similar problems may later be derived. Learning always imply some kind of generalisation. A fundamental difference in learning methods is whether the generalisation steps are performed when knowledge is stored (as in induction-based learning), or when problems are solved (as in case-based reasoning).

A fundamental principle for sustained learning is that each problem solving experience is a powerful source of learning. A necessary criteria for the learning to be successful is that a mistake once made will not be repeated. A role of the deeper model is to give the system a rich knowledge fundament for problem solving and learning: It will enable knowledge intensive case-based learning and reasoning by generating

[1]The development of CREEK is part of a Ph.D research - now in its terminal phase - supervised by Professor Arne Sølvberg.

explanations to justify reasoning steps, extracting relevant features, retrieving relevant cases, indexing cases and deciding which parts of a case to store and which previous cases to remove.

Figure 1 shows a basic scheme for sustained learning. The middle box illustrates - at an abstract level - the necessary steps: First, make an internal description of the problem, i.e. try to understand the problem by representing it within the system's internal knowledge model. Next, use whatever type of knowledge and reasoning method appropriate to produce a satisfactory solution to the problem. Finally, retain knowledge learned from this experience, so that the same or a similar problem may be solved with less user interaction, or more efficiently, in the future.

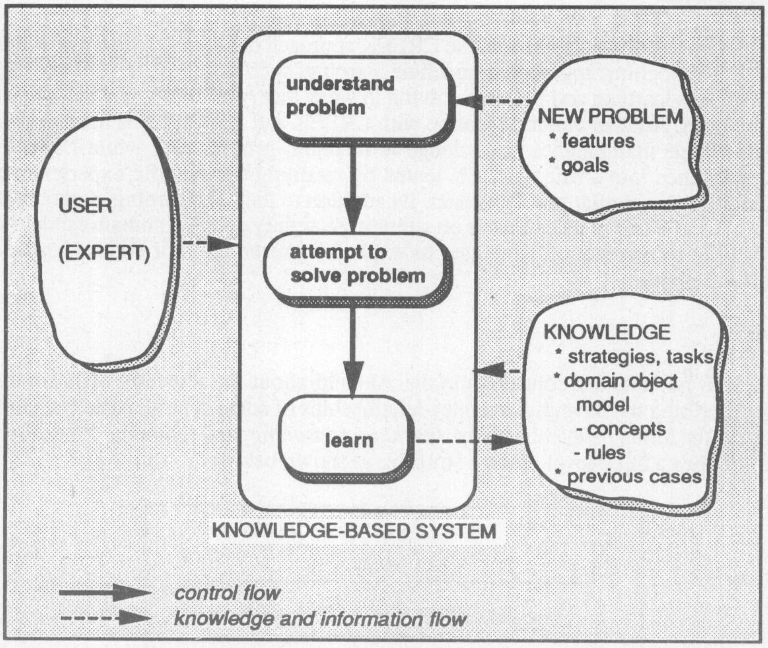


Figure 1: Sustained learning through problem solving
The figure shows a model of sustained learning. Knowledge - of different types - is used for learning as well as problem solving, and gets updated by the learning methods. The model incorporates user interaction both in problem solving and learning.

None of today's commercially available machine learning systems are able to learn continually from experience in real world environments; nor are classical induction methods like ID3 [27a], Version Space [22] and AQ [20]. Beside being unable to take advantage of an evolving knowledge base in their learning process, these methods also assume a classical view of concept definition. This view defines a concept as a set of attributes that are singly necessary and jointly sufficient to classify examples of a concept. Such definitions may be adequate for mathematical and other strictly formal concepts, but as shown by several authors [41, 33, 31] it is extremely difficult to present classical definitions of naturally occurring concepts like 'game', 'bridge', 'chair', 'car', 'bird', etc. Smith & Medin [33] distinguish between the 'classical', the 'probabilistic' (degree of prototypicality) and the 'exemplar' (purely extensional) view of concept definitions. Natural concepts are defined intensionally by a set of *typical* properties - i.e. a set of default properties - and/or extensionally as the set of individes, or exemplars, that are classified by the concept. Methods that try to learn non-classical concept definitions need support from an extensive domain model in order to explain why an exemplar is within the range of a concept, why some features are more relevant than others, etc.

2. An integrated framework

This paper describes the CREEK framework for supporting the development and maintenance of competent systems that learn from experience. The three properties listed below should be satisfied by the resulting systems. Hence these system properties defines the requirements of the framework:

- An expressive and extendible knowledge representation formalism, enabling an explicit and thorough modelling of relevant knowledge types.
- Problem solving and reasoning methods that are able to effectively combine reasoning from past cases with reasoning within a competent and robust model of more general domain knowledge.
- A learning method that is able to retain concrete problem solving experiences, and integrate each experience into the knowledge model in a way that gradually improves knowledge quality and problem solving performance.

Although differing in its case-based approach, the CREEK approach has some similarities with other systems that take a knowledge modelling approach to machine learning, such as the BLIP [23] and DISCIPLE [35] systems. They all address learning and problem solving in weak theory domains, and include user interaction in their learning methods. Another common feature with CREEK and DISCIPLE is the emphasis on learning as a continous knowledge maintenance (knowledge refinement) process. But while DISCIPLE learns by generalizing an experience into a rule, CREEK learns by retaining the specific experience in a form that enables it to be used to solve similar problems later. IN addition to the case learning method, CREEK differs from both DISCIPLE and BLIP in its emphasis on a thorough, tightly coupled domain model - enabled by an expressive and flexible representation language, its explicit representation of control knowledge, and its combined reasoning methods.

2.1. Basic notions

Unfortunately, there is no common consensus in the AI field about the meaning of basic terms. We will therefore start by describing the meaning and inter-relationships of some central terms frequently referred to in this paper, namely the terms *problem solving, learning, reasoning*, and *inference* The way the terms are used in this report implies a three-level process structure as shown below:

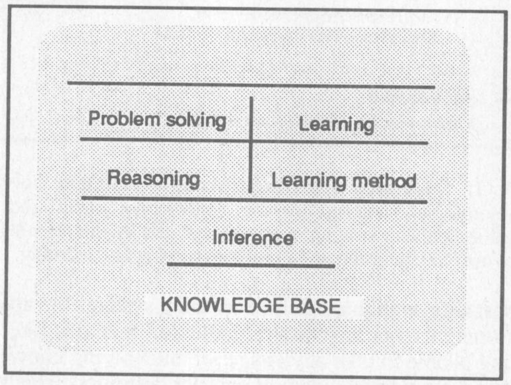


Figure 2: Structure of knowledge based processes

Problem solving is a process that takes a problem description, a goal, and a knowledge base as input, and derives a solution that satisfies the goal. The goal contains a specification of the requirements that must be fulfilled in order for a result to be a solution to the problem. A problem may be structured as sub-problems, and the problem solving process correspondingly split into sub-processes. The ability to solve problems is part of an agent's expertise, which includes domain knowledge, control strategies (plans), and reasoning methods.

Learning is a process that takes an existing problem solving system and new information as input, and derives a system where the quality and/or efficiency of problem solving is improved. To learn may, of course, be regarded as a problem in its own right, and problem solving could be said to subsume learning. To avoid confusion, problem solving will here solely refer to solving application problems.

Reasoning is a process that take some facts and a goal as input, and derives a result by applying one or more *inference* methods to a body of knowledge. Reasoning may be regarded as a more general term than problem solving, in the sense that its input and output may be any kind of information, not necessarily a problem description and a solution. Reasoning is also a sub-process of problem solving, since it is used to derive results that contribute to solving a problem. The term reasoning is most often used in connection with problem solving, and characterizes parts of the problem solving process, as in *rule-based reasoning, model-based reasoning, case-based reasoning*. Being a general process of deriving a result through inference, reasoning is also involved in some learning processes - and particularly in knowledge-intensive learning. In the machine learning literature, however, terms like *learning method* or *learning algorithm* are more frequently used when talking about particular sub-processes of learning, reserving the term reasoning to the problem solving process. This is also the view adopted in our work.

Inference denotes the lowest level of processes in the hierarchy, i.e. the building blocks for reasoning and learning. The inference methods define the primitive operations on the knowledge, and hence, the basis for its semantic interpretation. There are three basic types of inference processes:

- <u>Deduction</u>, which is a truth preserving inference process where an inference step typically is a logical derivation of a theorem from a theory that is assumed to be both consistent and complete. Deduction is based on a universal inference rule, like *modus ponens*. As an inference rule of first order predicate logic, deduction guarantees the truth of the conclusion given the truth of the premisses. But the requirements on the knowledge as a non-contradictory set of theorems are often hard to meet in real world contexts.

- <u>Abduction</u>, also called *inference to the best explanation*, is a kind of reverse deduction that preserves falsity, not truth. An abductive conclusion therefore has to be supported by an explanation that leaves the particular conclusion as the best choice. Abduction is also called *plausible inference* ([Charniack-84]), and is closely tied to generation and evaluation of explanations. Abduction is considered to be the kind of inference typically made in diagnosis (at a top level, at least) where a set of symptoms infers a fault by being explained by the fault, i.e. by generating a plausible argument that supports a particular fault and weakens competitive fault hypotheses.

- <u>Induction</u> is a process that synthesizes a general description (e.g. a concept definition) from a set of examples and, possibly, counterexamples. Induction is similar to abduction in that the truth of the premisses (the examples) does not guarantee the truth of the conclusion (the generalization). Therefore, an inductive conclusion should also be supported by a justification.

Deductive inference is a strong method, but its applicability for reasoning is limited by assumptions that do not generally hold for real world situations: A world - typically assumed to be closed - where all phenomena (all properties of concepts) are either true or false, and a consistent domain theory by which any proposed fact may be proved to be true or false. The advantage of inductive and abductive inference methods is that they are not necessarily subject to the limitations of first order logic: The world may be viewed more realistically as an open, dynamically changing environment. Concepts may be described in terms of typical features or default values which may be inherited to more specific concepts, and replaced by local values if these later become available. A disadvantage is that the particular inference methods needed has to be defined by the system developer, leading to a complex and often unclear definition of knowledge semantics. (The issue of deductive vs. non-deductive inferences and reasoning is given a deep and thorough discussion by the collection of papers in [18]).

Based upon the three *inference types* described, a range of *inference methods* have been developed in AI. An inference method specifies a way to derive new information from existing knowledge in an actual representation system. Examples are specific methods for chaining of rules, for inheritance of properties in network hierarchies, for matching of concept schemas, etc. Inference methods may be put together to form *inference structures* (like Clancey's inference structure for heuristic classification [10]).

2.2. A model of knowledge

2.2.1. Knowledge types

A thorough knowledge model of a domain contains different types of knowledge. In order to reason with various knowledge types, the system needs to know their different properties. That is, the system should contain an explicit model (meta model) of its own knowledge structure.

A given piece of knowledge may be classified by its position on a set of knowledge dimensions (figure 3). The following four knowledge dimensions reflects important semantic aspects of knowledge, from the perspective of knowledge-intensive problem solving and learning:

Knowledge *level* refers to whether the knowledge describes concepts of the application domain - called object knowledge - or the use and control of the object knowledge - called control knowledge. Knowledge *depth* refers to a scale from fundamental, theoretical, deep model or ´first principles´ knowledge, up to readily applicable rules of thumb or concrete past experiences that directly associate a solution with a set of problem descriptors. Knowledge *generality* classifies knowledge according to how large a space of the real world it describes. Conceptual model knowledge is typically general knowledge, since it contain class definitions, described by general relationships with other classes. A more shallow type of general knowledge is typically held within heuristic rules. A more specific type of knowledge is the knowledge contained in past episodes and concrete experiences, for example previously solved cases. Knowledge *role* is a dimension related to the use of knowledge. It splits into two subclasses: Descriptive knowledge - defining conceptual structures, stating established facts and assumptions - and operational knowledge, like a direct association between an input descriptor and a goal concept, or an action sequence that may be activated in order to reach a particular goal. The fifth dimension shown in figure 3 is related to the syntactic *form* of the knowledge, i.e. whether it is expressed as a concept network, if-then rules or units of cases.

Explicating these dimensions makes it possible to attack some of the major problems of today's knowledge based systems, by:

- Explicit modelling of problem solving and learning strategies, which enables a system to reason about strategies as well as object knowledge
- Combining shallow and deep knowledge, which may be present both at the control level and at the object level (although control level knowledge typically is shallow)
- Problem solving by combining previous experiences with more generally applicable knowledge
- Relating knowledge to its role in problem solving and learning, extending the notion of knowledge to include methods and procedures as well as concept schemas and heuristic rules

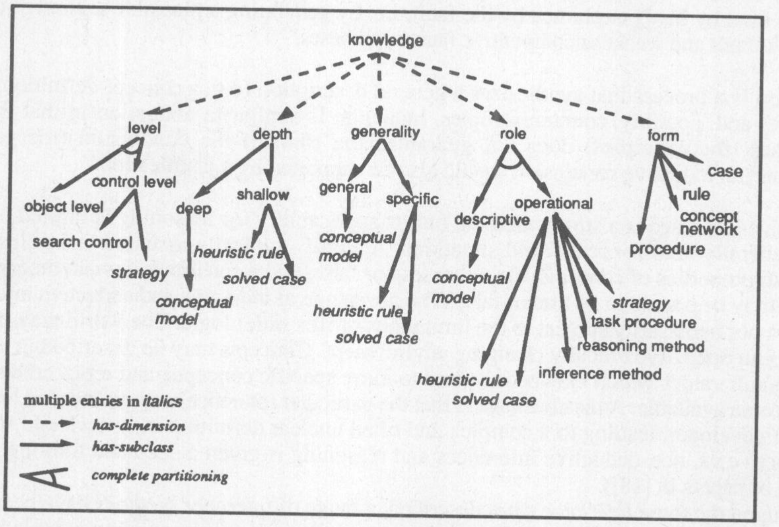


Figure 3: Dimensions of Knowledge
The four semantic - and one syntactic (form) - dimensions for characterizing knowledge, described by specializing them into subclasses.

2.2.1. The knowledge network

The term *knowledge model* refers to the total network of concepts and relations of different types - including heuristic rules - as well as the collection of past cases. The nodes in the network are concepts (entities as well as relation type concepts), and the links are relations. A relation is also a concept with its own definition. Cases and rules are coupled to the conceptual network model by having all their descriptors explicitly defined in the model.

A knowledge model should be viewed as a single, tightly coupled network of concept definitions, heuristic rules, past cases, procedures, methods, and strategies. The basis for the system's understanding is the fundament of descriptive knowledge formed by explicit definitions of all terms used in the various submodels.

2.2.2. A model of expertise

When the 'traditional' notion of knowledge is extended to incorporate reasoning methods and problem solving strategies as well, it is often referred to as *expertise* (e.g. [40]). Human learning is to a large extent an automatic mechanism, but experts also acquire *knowledge about how to learn*, which may also be used deliberately to improve learning. This knowledge is used together with domain knowledge to decide, for example, which parts of an experience are worth remembering, and whether some feasible generalisations should be made.

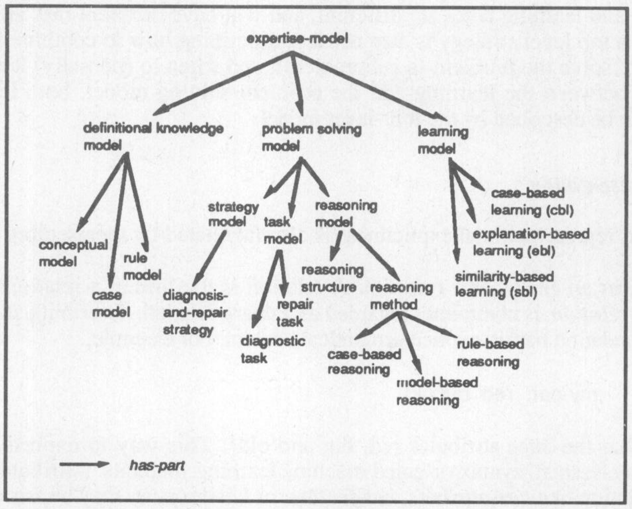


Figure 4: A Component Model of Expertise
The figure shows components of the expertise model, with control knowledge for diagnosis-and-repair type of problems.

The extension of the knowledge concept towards expertise, in the sense sketched here, is a consequence of recognizing problem solving and learning methods as knowledge to be modelled and represented, just as factual, object level domain knowledge. In addition to the structural component model of expertise shown in figure 4, a functional model - a methodology for integrating and controlling the submodels - is needed. In the KADS project [6, 40] an approach to modelling of problem solving expertise has been developed, based on four layers of expert knowledge: object domain layer, inference layer, task layer, and strategic layer.

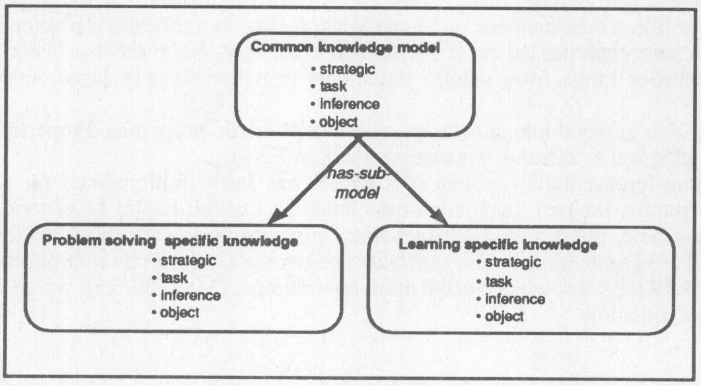


Figure 5: Integrated Four-Layered Expertise Model
The KADS four-layer model may be used as a basis for modelling learning competence as well as problem solving competence by integrating separate problem solving and learning knowledge into a common knowledge model.