RMIT

THE CONCEPT OF KNOWLEDGE ENGINEERING AND EXPERT SYSTEMS FOR COMPUTER PROFESSIONALS

JACOB L. CYBULSKI
Department of Computing

Royal Melbourne Institute of Technology
GPO Box 2476V, Melbourne 3001, Victoria
Australia

Technical Report 84/4

The Concept of Knowledge Engineering and Expert Systems for Computer Professionals

Jacob L. Cybulski
Department of Computing,
Royal Melbourne Institute of Technology.
GPO BOX 2476V, Melbourne 3001, Victoria.
AUSTRALIA

Abstract

The aim of this paper is to provide computer professionals with a comparative analysis of the classical computer systems (eg.: Data Base Management Systems) and the new generation of systems (eg.: Expert Systems and Knowledge Based Systems).

The paper treats various aspects of data base management and software development techniques, and shows that Expert Systems are a natural continuation of currently used commercial systems. Attention is focused on the problems of information representation, information acquisition, information processing and management, and finally on different techniques of assisting the user.

Categories and Subject Descriptors : [Artificial Intelligence]: Knowledge Representation Formalisms and Methods, Expert Systems.

Introduction

Today's computer users are becoming more and more fastidious in dealing with computer hardware and software. They are not satisfied anymore with computer tools which merely solve their problems. They expect a computer system to be simultaneously easy to use, flexible, robust and efficient. These goals are very difficult to achieve concurrently, and there are very few software products which meet these requirements.

User-friendliness (ease of use) can be achieved in several different ways, via:

- the construction of new, better computer interfaces;
- storing and analysing human-like, natural knowledge;
- problem solving support;
- the active role of a computer in a man-machine interaction.

Research over the last few years has been very fruitful in providing new well-designed products for individual computer customers. The user interface to the machine has been greatly improved with graphic packages (windows, forms), voice synthesisers and analysers for personal computers, and natural language interfaces for the whole range of machines from mainframes (INTELLECT), mini-computers (THEMIS), to micro-computers (ASK and Symantec newly announced products) (Davis 1984). Relational data base management systems and the new generation of database machines offer the promise of easier manipulation of complex data. Furthermore some specialised application packages like spreadsheets (1-2-3, MicroPlan, etc.), and the fourth generation languages facilitate problem-solving for business and industry. Finally the Japanese Fifth Generation Project aims at providing a new machine based on new principles (a PROLOG-like language as the fundamental programming tool), which will permit problem-solving in an easier, more natural way (Robinson 1983).

Package flexibility is a feature which is more difficult to achieve than ease of use. Some problems in software flexibility have been resolved. Research could now be directed to the following problem areas:

- handling unrestricted queries;
- multi-problem solving;
- handling incomplete and unreliable data;
- representing beliefs.

Development of natural language interfaces, which can be regarded as improving the variability of user queries, has gone a long way towards dealing with the first problem area; however the diversity of possible questions which could be asked of the system must also be reflected in a structure of the data base in use, so that we have to go beyond the user/database interface. The second problem area arises from programs' inability to adjust to new conditions or new problems even from the same application area. Research in the remaining two areas aims at program competence in efficient handling of uncertain information - where fuzziness is caused by incompleteness of the data base, unreliable sources of information, or the system analyst's beliefs. Few of the classical computer systems address any of these four problem areas.

We now come to the last two system qualities mentioned - robustness and efficiency. Sometimes they are disregarded by academics, who concentrate on the actual solubility of a particular problem rather than on the implementational aspects of the solution; however this situation is not welcome in such practical fields as business or industry.

The lack of computer robustness and efficiency is no longer a question of physical devices; it is an obstacle which has to be dealt with by problem analysts, and software designers. And, as with many other areas of a computer use, there is a shortage of automated tools which could help us to resolve problems efficiently.

The discussion so far indicates that there are gaps in computer technology, gaps which must be analysed on purely theoretical grounds and solutions then implemented practically as either hardware or software.

Attempts to address some of the above issues have been made by researchers on the New Generation of Computers in Japan, USA, and Europe. The ultimate answers have not been found, but a few successful software products, aimed at the future generation of computers, have already appeared at the open market. These new packages are often a result of combining various scientific fields previously considered unrelated, such as computer science, mathematics, psychology, neurology, and linguistics. The concepts of information processing have changed, and the new generation systems do not process mere <u>data</u> (pure information), but human-like <u>knowledge</u> (information + its use); the ultimate goal is to

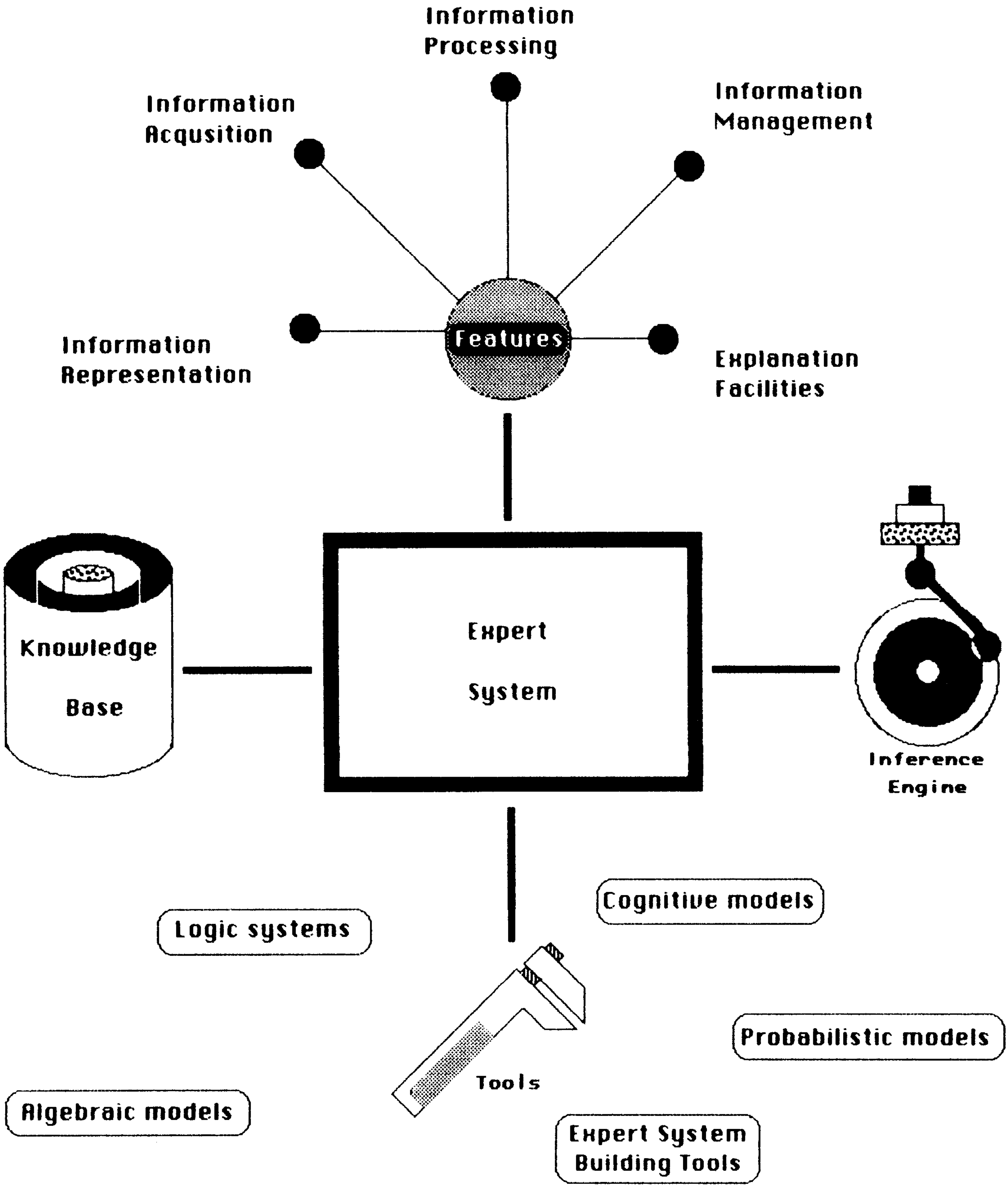


Figure 1 - Various aspects of expert systems

represent wisdom (information + use + judgements about information and the way it is used). Now we can understand why knowledge of human cognition (knowledge processing - psychology), brain structure (information storage - neurophysiology), formal theories of knowledge and language processing (modelling - mathematics + linguistics) are so important to the new computer systems.

Complex problems require the use of knowledge (or wisdom), therefore automated complex problem solvers will have to provide a general schema for representing knowledge in the computer. The total knowledge represented in a machine will be refered to as a knowledge base (KB), and the system using and aquiring knowledge will be referred to as a knowledge based system (KBS). A knowledge based system which shows the performance comparable to that of a human expert in the field is called an expert system (ES).

Knowledge based systems and expert systems are those computer tools that we expect to fulfill our expectations about user-friendliness, flexibility and high performance. The following sections will draw attention to these systems and the facilities they offer.

Facilities offered by Expert and Knowledge Based Systems

Expert and knowledge based systems have significantly different goals to traditional computer systems and therefore they offer different facilities to the user. The aspects of information processing we would like to talk about are as follows :

- information representation;
- information acquisition;
- information processing;
- information management;
- assistance to the user.

Information representation

Most of the expert system characteristics which are explained in this section result from the characteristics of the objects the systems deal with. The substance subjected to the processes of knowledge based systems is knowledge, therefore we must have appropriate models to describe and manipulate information and its use (see introduction for the definition of knowledge and wisdom). To use traditional terminology we have to be able to store and process both data from a particular domain and application programs solving problems from this domain. To project further into the future, ultimately we will be looking at wisdom-based systems, where we will need appropriate models to make judgements about information and its usage.

This leads to the first disparity between classical and new computer systems. The old systems were divided into two separate conceptual entities: programs and data. In the new systems, both programs and data are embedded in the knowledge base. This is not a new concept in data processing. The revolution in computer architecture started with von Neumann's concept of the stored program, which is interpretively executed, statement by statement; thus it is treated like data (Goldstine 1972). However while data processed by a particular program is given meaning by this program, the meaning of the program is inherent in its own behaviour, and we do not have any simple means to describe a von Neumann program's behaviour other than by executing it. The new systems aim to provide methodologies and tools enabling a description of conditions, structures and goals of programs kept in the knowledge base. This unified approach enables us to reason about programs as well as about data.

Information acquisition

Building a database is a difficult task. First of all we must find all possible data types occuring in the sphere of discourse; then we have to analyse and categorise all the kinds of natural relations between the data types; and then we construct a generalised schema for the database which is supposed to cater for all possible information which can be obtained from that sphere.

Writing an application program is an equally difficult task, which demands full attention to a number of problems requiring solution, a detailed analysis of relationships between data, databases, and operational programs, and also - what is probably the most laborious part of the job - demands interaction between domain experts (like doctors, lawyers, etc.), future system users, and finally the machine.

Constructing a knowledge base which interweaves data, knowledge of data (meta-knowledge), and sometimes knowledge of data models in use (schema-schema) is a much more difficult activity. Here we must combine both expert level information (data) with the knowledge of how it should be used (program). And because there is no physical division between these two entities, the creation, testing and correction of the knowledge base must be constantly supervised by experts and users. The transfer of expertise between men and machines must be controlled by a skillful system analyst, sometimes referred to as a knowledge engineer. Knowledge engineering is a responsible job, which could be eliminated if an appropriate tool were provided, enabling a direct transfer of expertise from the domain-expert to the knowledge base. This facility would have to test and exercise the knowledge kept in the system, and the expert should be able to enter corrections to data, data structure and programs. The existing expert systems provide such a knowledge acquisition facility, however it is usually a domain-specific tool, which is hard to transfer to another area of expertise (eg. in medicine - IRIS, Trigobow and Kulikowski 1977). Some attempts to create domain-independent knowledge-acquisition tools for particular knowledge bases have been successful (eg. EXPERT, Weiss and Kulikowski 1979; GUIDON, Clancey 1979); another direction of research was to create universal knowledge acquisition systems independent of a specific domain, and knowledge representation (TEIRESIAS even though used with MYCIN employs independent techniques - Davis and Lenat 1982).

Information processing

The structure of data used by classical computer systems is more or less static, and depends primarily on the database schema (let's disregard the program's internal data structures). The processes using these data structures are deterministic, and mostly employing algorithmic, and well-defined solutions to problems. As we all know, complex information, like that processed by a man, has a completely different nature. It is dynamic, ill-structured, patchy, fuzzy, time and situation dependent, etc... The only conclusion, then, is that we cannot use the same techniques for data and knowledge (or wisdom) processing.

As opposed to data processing using algorithmic reasoning (a well defined sequence of events), knowledge processing is based on heuristic reasoning. The inference engine - or problem processor - is looking for partial solutions to a given problem, proceeding by trial and error; it is applying heuristics ("rules of thumb") stored in the knowledge base, expecting that either the answer will be found or that more evidence will appear. The sequence of actions is not forseeable, since many factors determine which of the rules are chosen during the problem analysis. The possible factors are: query contents, query form, knowledge base state, internal conditions, etc...

An inference engine can use two different modes of processing. One is exact reasoning, where an extensive, but heuristically controlled search of the data base is done. This method of operation requires a reliable source of information, and a complete knowledge base (full information on the problem is stored). The advantages of exact reasoning can be immediately found in scientific and engineering areas where all possible data is usually available (sometimes it is a huge amount of data), and all possible rules and formulae are well known. Figure 2 shows a session between the exact reasoning system (bold face) and a user (plain text; for more examples of exact heuristics see Davis and Lenat 1982):

1. Text:	John was teaching all the day yesterday.
2. User:	What was John doing?
3. System:	**John was teaching.**
4. User:	When was John teaching?
5. System:	**John was teaching yesterday.**
6. User:	What time did John finish work yesterday?
7. System:	**Sometimes after 5pm.**
8. User:	Where was John yesterday?
9. System:	**John was at school.**
10. User:	Who did John spend all the day with yesterday?
11. System:	**John spent all the day with his students.**
12. User:	List all the facts used in question answering.
13. KB facts:	**Teachers teach.** **Teachers work at school.** **Teachers work from 9.10am to 5.30pm.** **Teachers teach students.**

Figure 2 - A session with exact reasoning system

The example shows the power of heuristic resoning. First of all the system can act like a normal data base retrieving known facts (lines 2 and 4), secondly it can provide facts it does not know explicitly but which can be deducted from the text in question and the knowledge stored in the knowledge base (lines 6, 8, 10; knowledge base facts 13).

The exact reasoning utilized here restricts the format of internal rules used to answer all possible questions; they can exhibit a typical pattern of: the **GOAL** is true (say - the answer is found), providing that **CONDITIONS** are fulfilled (say - there exist a chain of facts leading from premisses to the answer). This pattern, similar to an "**if-then-else**" statement, reminds us of an algorithmic approach to solving problems; however the heuristic rules are related one to another only via their domain of application; their use, ordering and scope of their application, which in traditional terms is known as a program structure, is unknown and is determined by an inference engine at run-time. This implies that to create a knowledge based system we add to it a set of unrelated facts and rules in some appropriate format, and they will be sorted, analysed and then used automatically (this is obviously over-simplified, but it is an idealistic principle we would like to achieve).

Some interesting applications of exact reasoning can be found in operating systems configuration (R1 - McDermot 1981), VLSI design (EURISKO - Lenat, Southerland and Gibbons 1982), or in constructing intelligent terminal agents (RITA - Anderson and Gillogy 1976). The typical interaction with such a system consists of providing definite answers and values to the program. In practical terms, this is not always possible. Some values cannot be given unless expensive measurements are performed (as in geology), or unless harmful experiments are completed (eg. in medicine). In such cases exact reasoning would be impossible and of no value to the expert system.

Another mode of operation is called inexact reasoning, which enables users to respond to system queries by providing degrees of confidence in the data stored in the knowledge base; therefore users can refer to their experience, beliefs, estimates, and intuition. Systems employing this type of methodology frequently use probabilistic or possibilistic data and reasoning models (see Mamdani and Gaines 1981). A typical session is shown in figure 3 (system text is in bold face, user's in plain text).
This particular session illustrates two different features of natural knowledge which cannot be dealt with exact reasoning systems. One of the features is data uncertainty (text 1, 2, 3), where degree of user's confidence in some facts is not 100%; the second is results' ambiguity which can be solved by varying confidence in different answers deducted automatically from the text on input and the knowledge base (lines 4÷11).

The best application areas for inexact data models are all those areas where knowledge cannot be easily verbalized (in terms of logic, event sequence etc), or where the user-machine interaction is imprecise for instance in fault analysis (AL/X - Reiter 1980), illness diagnosis (MYCIN - Shortliffe 1976), mineral prospecting (PROSPECTOR - Duda, Gashing and Hart 1979), or even language understanding (PRUF - Zadeh 1977).

1. Text:	[080%] John was studying yesterday.
2. Text:	[050%] John stayed at home yesterday.
3. Text:	[100%] John is a programmer.
4. User:	What was John doing?
5. System:	**[060%] John was studying.**
6. User:	When was John studying?
7. System:	**[062%] John was studying yesterday.**
8. User:	Why did John stay at home yesterday?
9. System:	**[030%] Because yesterday was a weekend.** **[030%] Because John took some time off for studying.** **[030%] Because John was sick.** **[010%] Because John has a flexi-time and does not need to come to work every day.**
10. User:	List all the facts used in question answering.
11. DB facts:	**[090%] Students are studying.** **[095%] People don't work during weekends.** **[085%] People stay at home when sick.** **[040%] People working on flexi-time can work when and where they want to work.** **[030%] Programmers very often work flexi-time.**

Figure 3 - Examplary session with inexact reasoning system

(Degree of confidence given in square brackets)

Information management

The outcome of knowledge engineering is a set of well-defined facts and rules about a particular domain. The facts can be viewed as data elements and records, the rules as relationships between data elements, records and paths. Therefore a set of knowledge base rules describes the knowledge base logical structure - the equivalent of a data base schema. In spite of the similarities between data and knowledge bases, there are some distinguishing features.

The first difference comes with the acquisitive character of knowledge bases, which are capable of acquiring not only data, but also their structure and knowledge of how to use the data and their structure from the knowledge engineer (structure=schema and use=application). Classical databases are usually restricted to a single data model, and also to a single data base schema; any changes in the data structure (via a data base management system) will affect not only the data base itself, but frequently application programs as well. This does not happen with knowledge base systems, which have a much more dynamic character.

The second contrasting feature can be found in some more advanced knowledge base building tools. The new feature is a so called schema-schema, which "provides a foundation of representation - independent knowledge that can be used for constructing an entire knowledge base" (TEIRESIAS, Davis and Lenat 1982). This gives enormous flexibility in knowledge manipulation, and subsequently in knowledge engineering. Thus on the one hand the knowledge engineer can freely experiment with data, data structures, and also data models; on the other hand building and testing expert systems becomes a more complex task to perform.

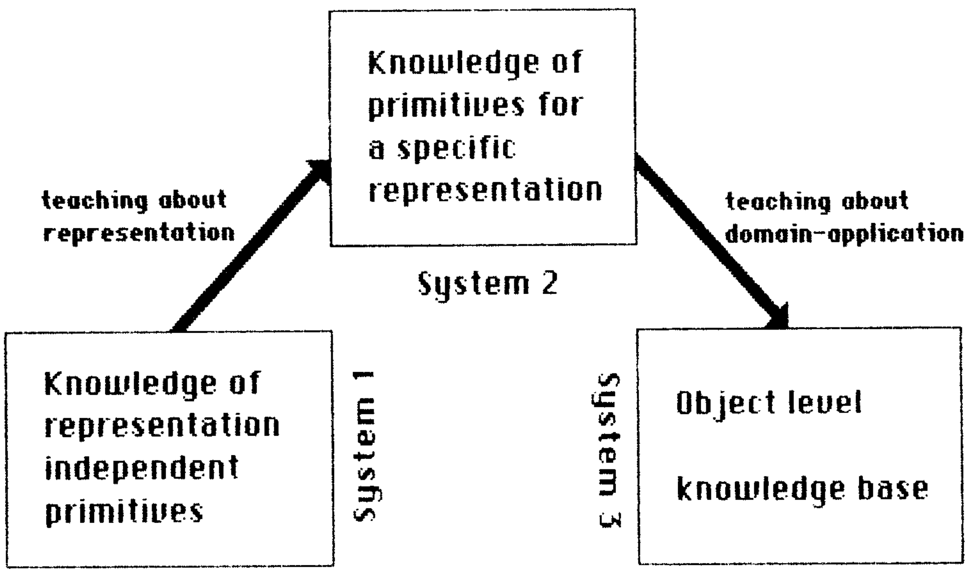

Knowledge of primitives for a specific representation
teaching about representation
teaching about domain-application
System 2
Knowledge of representation independent primitives
System 1
System 3
Object level knowledge base

Figure 4 - Schema-schema architecture
(from Davis and Lenat 1982 p 392)

The use of a knowledge base in the schema-schema architecture requires three conceptually distinct activities: the creation of an appropriate data model independent from any application (System1 - n.b. - Davis' terminology is "representation" rather than "data model"), then teaching the system about a particular domain of knowledge (System 2), and finally we can use the knowledge base system to solve our domain-specific problems (System 3). Some systems permit intermixing all these stages of work.

<u>Assistance to the user</u>

Finally let us concentrate on another aspect of user-friendliness we haven't spoken about.

Both data base systems and knowledge base systems are very complex systems of information processing, and therefore should have facilities to assist the user in data base creation and data manipulation. The usual aids given to the users are system prompts (with explanations what to do next), menus (with explained alternatives), or help functions (enabling a detailed explanation of particular commands, etc...). A well-designed data base management system will also provide the means to list the data base structure, and its contents.

Knowledge base systems give something more: explanation facilities. They offer the tools able to answer old-fashioned questions: what and where? And also the tools which answer the questions: why and how ?

Figure 5 shows multilayered explanations in an expert system identifying gemstones (system text in bold face, user responses in plain text; for other examples of explanation facilities in say medical expert system see MYCIN, Clancey 1981). Users of expert systems in fields such as medicine, law, or geology (the favorite expert systems' fields) have a moral and professional responsibility not to accept the system's advice, unless a full explanation is provided.

Similarities and differences between classical computer systems and expert systems are summerised in figure 6.

system - **bold type**
student - plain type

Student Identify the gemstone.

MGCAI
locality: unknown
form: stone
size: small
weight: 400mg
colour: yellowish
lustre: dull greasy
habit: what?

MGCAI
apparent shape of the crystal.
habit: octahedral
streak: grey
hardness: why?

MGCAI
Your answers indicate a possibility of the following stones:
diamond,
scheelite,
senarmontite,
others too.
Hardness of the stone can be decisive.
hardness: how?

MGCAI
Use another mineral and try to scratch the stone with it, result given in Moh's scale:

1 (talc),	**2 (gypsum),**
3 (calcite),	**4 (fluorite),**
5 (apatite),	**6 (orthoclase),**
7 (quartz),	**8 (topaz),**
9 (corundum),	**10 (diamond).**

Instead of standard minerals you can use:

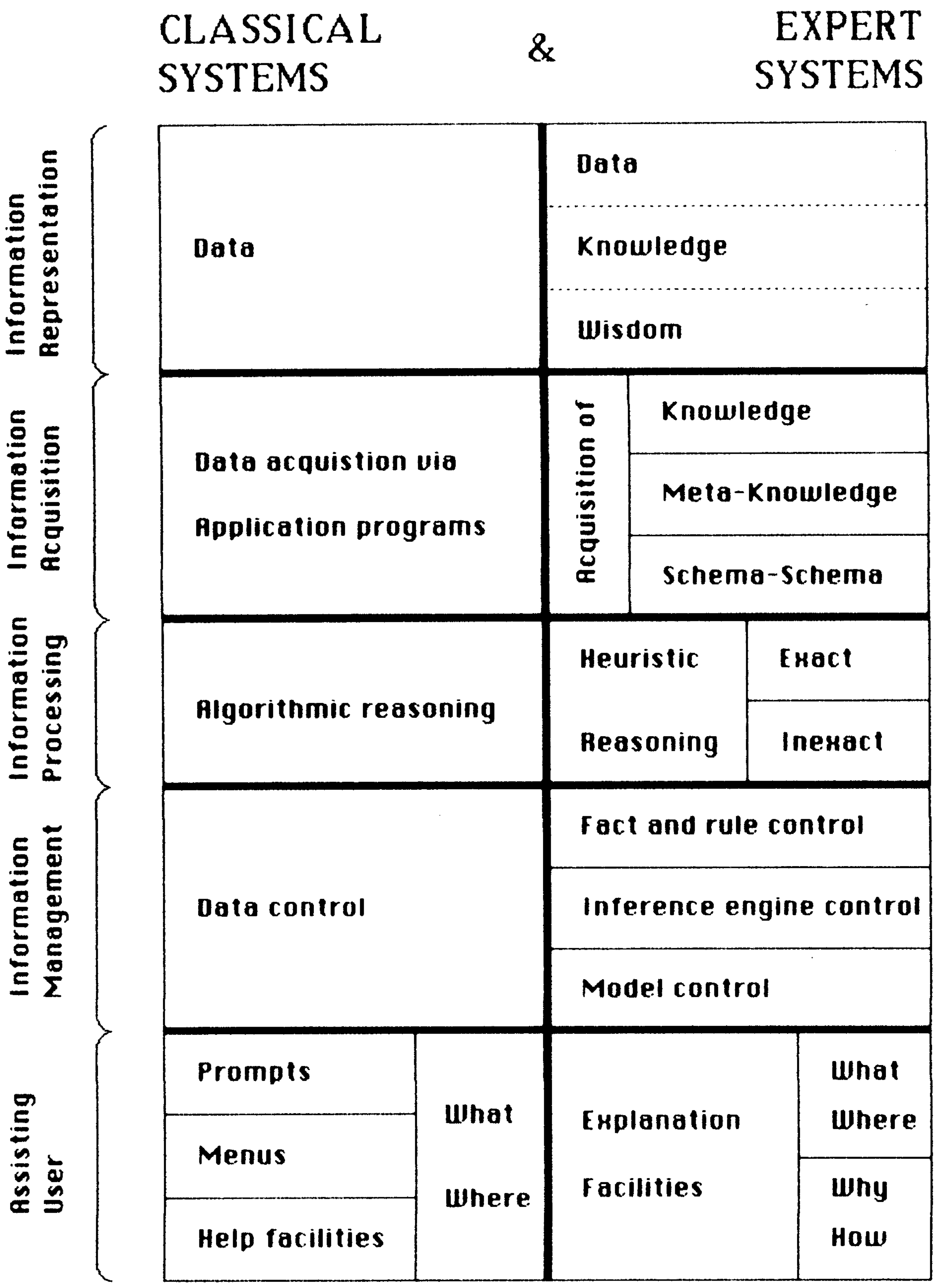
2 (fingernail),	**3 (coin),**
5 (steel knife),	**6 (steel file).**

hardness: more than 5

MGCAI **Everything indicates it is a diamond.**

Student Why ? ...

Figure 5 - Explanation facilities in expert systems



	CLASSICAL SYSTEMS		EXPERT SYSTEMS	
Information Representation	Data		Data	
			Knowledge	
			Wisdom	
Information Acquisition	Data acquistion via Application programs		Acquisition of	Knowledge
				Meta-Knowledge
				Schema-Schema
Information Processing	Algorithmic reasoning		Heuristic Reasoning	Exact
				Inexact
Information Management	Data control		Fact and rule control	
			Inference engine control	
			Model control	
Assisting User	Prompts	What Where	Explanation Facilities	What Where
	Menus			Why How
	Help facilities			

Figure 6 - Features of classical and expert systems

Architecture of expert and knowledge based systems (ES & KBS)

The basic KBS-architecture is similar to that of a data base system. Analogously to the communication with data base systems the user interacts with the system via a query language which can be of an arbitrary form such as speech waves, written English, graphical icons and others. The query language is translated into a data manipulation language, operating directly on the knowledge base (as compared to a data base).

The core of the system is a knowledge base, which consists of facts and rules (as opposed to data and schema). The user's problem embedded in a question or a command is processed by a dedicated general problem solver inference engine which corresponds to the application program and which is the reasoning centre; the correspondence is of an organizational rather than conceptual nature, and the inference engine's architecture and behavioural patern bears little resemblance to the conventional model of computation. We must also remember that the major part of the application knowledge is stored in the knowledge base rather than in the inference engine.

Figure 7 shows a data flow from the user through the inference engine (a query language agent must mediate between them) to the knowledge base (data manipulation language must have its own agent), and back. The elements worth further discussion are: the knowledge base and the inference engine. Let us take a generalised approach.

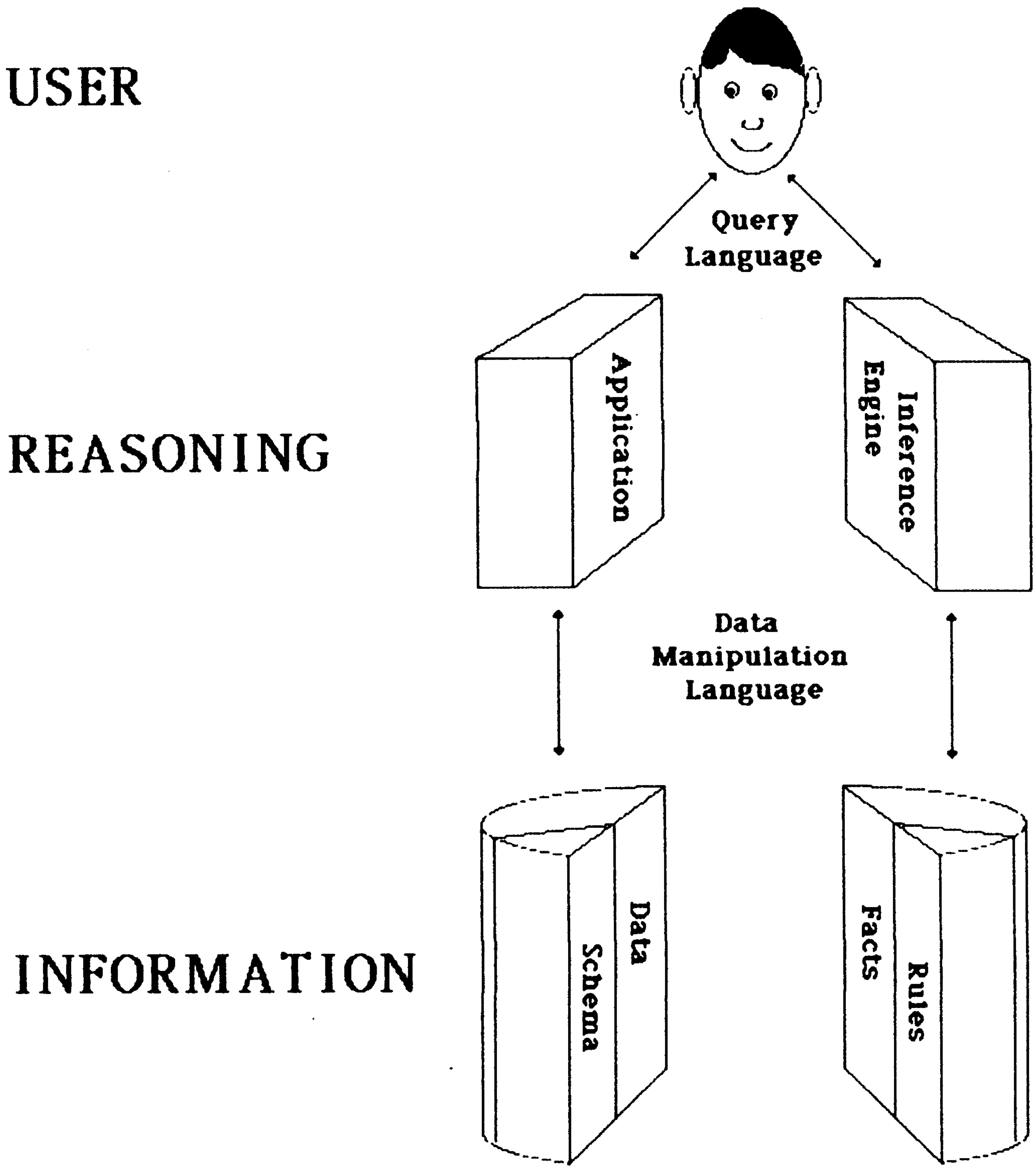


Figure 7 - Data flow in DBS and KBS

Knowledge base

The previous sections contain brief descriptions of this information store. Let us elaborate a little bit more on the contents of a knowledge base (KB). As we have already mentioned it consists of facts and rules, and they can be categorised into the following classes:

- facts and rules about the world
  - engineered knowledge,
  - user supplied data;
- facts and rules about the self
  - meta-knowledge,
    - domain dependent inference rules
  - schema-schema,
    - domain independent inference rules;
- facts and rules about interactions
  - knowledge of query language,
  - knowledge of user-machine protocol,
  - knowledge of the user.

Facts about the world is the user-supplied data or engineered information specific to an application domain. Rules about the world are also data-related and construct all valid relationships between facts and between the rules themselves (see figure 8).

The KB's knowledge about itself (facts about the self + rules about these facts) fulfills a double role: that of a data dictionary and that of a mechanism for meta-knowledge and schema-schema (see earlier discussions of these terms and also refer to the example in figure 8). Therefore the KB's knowledge about itself describes the rudimentary data structures used to represent knowledge and also outlines the fundamentals of the reasoning model assumed by an expert system. Facts and rules about inference processes (inference rules) can be of two different types: domain-independent and domain-dependent. The first ones are used to govern higher level reasoning, which cannot be achieved by mere use of the inference engine (which enables very primitive reasoning only). The latter introduces a more efficient processing of information about a specific area of knowledge; it provides a set of short-cuts and simplifications, which would not be formally permitted outside of this specific domain of application. The inference rules define an order in which facts and rules are analysed, the depth and extent of the data base scan in the process of query processing (the real time interaction for example does not allow an extensive search for data), etc.

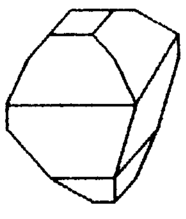
A number of facts and rules enabling the physical communication between a user and the knowledge base is also required (this could be thought of as knowledge about the query language and the data manipulation language - facts about interactions). They can describe user-machine interface protocol, language grammar, the user psychology, etc (see figure 8).

The implications of this organisation are clearly visible. All objects residing in the knowledge base can be manipulated by an inference engine, and therefore all data (the world knowledge), schema (the self knowledge) and application programs (domain-dependent inference rules) are dynamic and flexible (however this blend of data and programs leads to a much higher complexity of knowledge engineering tasks).

Knowledge of the world

Facts about TOPAZ:

formula	Al2(SiO4F,OH)2
system	orthorhombic
habit	prismatic cristals
colour	yellow or white
lustre	vitreous
streak	colourless
hardness	8.0
SG	3.4-3.65



Rules about TOPAZ:

if locality of finding is Brazil and stone was heated then allowed colours are pink & rose;

if locality of finding is Australia then allowed colour is none & blue

if habit is columnar or granular masses and most of other conditions apply then it is still TOPAZ

Knowledge of the self

Domain independent

if most of the individual facts comply with the prototype then the object has been identified;

if the user asks "what" then use the dictionary description of the most abstract term in the last sentence;

if the user asks "why" then interpret the inference path constructed during the concept recognition;

Domain dependent

if the formula, system and habit of the cristal are known then refer to the rock and mineral table for the direct answer;

if the user supplied data is insufficient to identify the gem ask the user to perform experiments which could yield further gem characteristics;

Knowledge of the interactions

Synonim and similar words classified by a grade of membership

Term	**Synonim**	**Membership**
WHITE	Pure white	1.0
	Greyish white	0.9
	Bluish white	0.9
	Yellowish white	0.8
	Whitish grey	0.7
	Grey	0.4
	Colourless	0.4
	Reddish white	0.3
	Greenish white	0.3
	Ash grey	0.2
	Shade of white	0.1
	Others	0.0

Term	**Synonim**	**Membership**
HARD	10.0	1.0
	Diamond hard	1.0
	Very hard	0.9
	9.0	0.9
	Hard like hell	0.8
	8.0	0.7
	Steel hard	0.6
	5.0-7.0	0.6
	Stone hard	0.4
	Not very hard	0.3
	0.2-4.0	0.3
	Not very soft	0.2
	Soft	0.0
	Others	0.0

Figure 8 - facts and rules in the knowledge base

Inference engine

The inference engine is a procedural part of the system while the knowledge base has a declarative nature. The engine consists of a set of universal knowledge-processing methodologies relevant to a specific inference model (eg. the resolution and unification principle - Robinson 1979, see also the Dempster-Shafer theory of evidence - Barnet 1981, or simple probabilistic Bayes rules). The inference model must be representation oriented and therefore the logic representations use theorem-proving techniques, procedural representations use different data and control flow methods, semantic networks utilize graph-theoretical or cognitive models of reasoning, production systems specialize in procedural activation, etc. Some other knowledge models combine a set of different representational schemata and therefore have hybrid types of inference engines.

The inference engine is the element of an expert or knowledge based system which is the hardest to implement, for it must be based on a well-defined rather than ad hoc formalism, and it also plays the roles of a system manager, scheduler, garbage collector, and so on. It is virtually a whole operating system.

A diagramatical comparison of data base systems and knowledge based systems is included in figure 9.

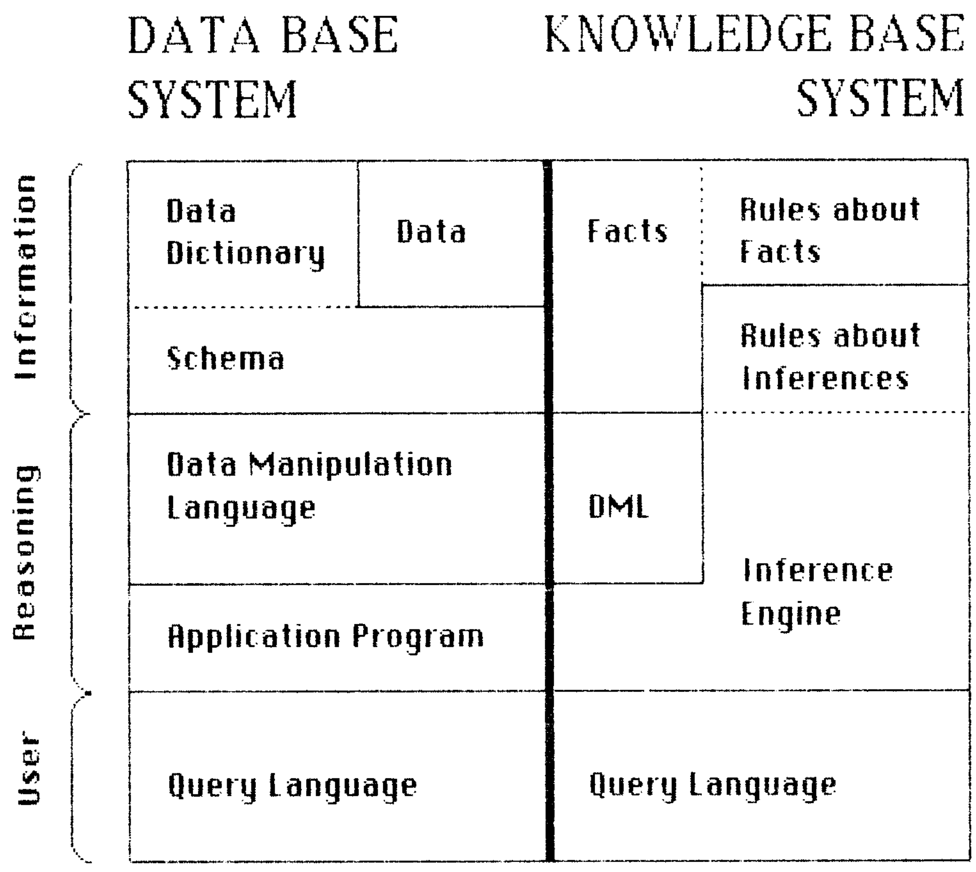


Figure 9 - DB and KBS architecture

Building Expert Systems

As we have noted the Knowledge Base Systems are of a significantly higher complexity than normal Data Base Systems. The rules of the Knowledge Base do not aim at describing merely the data binding, storing order, or access paths; they define complex relationships between various concepts, their syntactical, semantical and pragmatical values, and finally their plausibility. Such organization requires a much stronger formalism, than simple data base models such as Relational, Network or Hierarchical. More sophisticated tools are used enabling the utilization of a less strictly-typed knowledge, which ultimately leads towards an expressive power comparable to that of a man.

Several different model types are considered to be useful. Here we should mention logic systems, algebraic models, probabilistic and finally cognitive models.

The research in knowledge based systems (KBS) started around 1960 with the development of LISP (McCarthy 1961, 1962; Siklossy 1976) a specialised artificial intelligence (AI) language enabling easy construction and processing complex data structures. Other similar languages have been developed later on: QLISP (see QA1, QA2, QA3, QA4 - Green 1969, Rulifson et al 1971), SAIL and LEAP (Feldman et al 1972), POP-2 (Burstall, Collins and Popplestone 1971); then more sophisticated tools combining strict mathematical formalisms with programming concepts: PLANNER (Hewitt 1971), CONNIVER (Sussman and McDermot 1972), FUZZY (LeFaivre 1977); and finally a language of the japanese Fifth Generation project: PROLOG (Kowalski 1979, Clocksin and Mellish 1981).

It is not only the expressive power of data structures in the new languages that makes them so attractive, but it is also an entirely different approach to describing program control. Most of the new languages are of an interpretive nature and this is why programs and data are interchangeable. This has led to the idea that control-flow can be directed declaratively as well as procedurally (eg. PROLOG). And therefore in some of these languages the problem can be resolved automatically from its description, rather than by supplying a step by step algorithmic solution (i.e. descriptions are interpreted as programs).

It is clear that before we solve a problem algorithmically (the "old-fashioned" way), we must provide its specifications. The modern programming techniques (using expert system building tools, or languages) allow us to avoid the second phase of program construction, and to concentrate on a detailed problem analysis. Thus new AI languages permit the building of complex knowledge bases while focusing attention on problem solving, rather than on implementational aspects of the system. These languages help in designing knowledge base tools; however this does not solve all the problems.

Early research in artificial intelligence was dominated by attempts to develop general-problem solvers (Newell and Simon 1963, Ernst and Newell 1969). These systems were based on a few powerful heuristics, and the set of knowledge base rules was very limited. Soon it became evident that this approach in general was circumscribed by very small application domains (like Simon's wandering ant - Simon 1969). The answer to the weaknesses of these early systems was adding an extensive knowledge base of many rules and facts about a particular domain. If we consider the architectures of some very successful expert systems like

PUFF (providing expert analyses at a California medical center), or PROSPECTOR (which discovered a molybdenum deposit whose expected value exceeds $100m) (Hayes-Roth, Waterman and Lenat 1983), we can see that it is not an elaborate theory (applied to the inference engine) which drives the system, but an extensive and intensive knowledge codified into the knowledge base by domain-experts. This leads to the conclusion that the construction of appropriate tools for direct manipulation of the knowledge bases by experts is necessary (an example of such a tool is TIMM - Kornell 1984, Hunt 1984).

Another means of building expert systems or knowledge base systems are <u>expert system building tools (ESBT)</u> - the counterpart of data base management systems. These packages overcome difficulties in understanding the complex strategies and methodologies that can be employed by expert systems (like logical or probabilistic models), and give a set of interactive tools to build large knowledge bases (their schema).

Historically ESBTs evolved from particular expert systems, after the methodologies and tools applied had been extended and generalized; for example: EMYCIN evolved from MYCIN (van Melle 1980, Shortliffe 1976), Meta-DENDRAL from DENDRAL (Buchanan and Feigenbaum 1978, Lindsay et al 1980), KAS from PROSPECTOR (Hayes-Roth, Waterman and Lenat 1983, Duda et al 1978), HEARSAY-III from HEARSAY-I (Balzer, Elman and London 1980). The applicability of these tools depends on the domain of knowledge we want to represent in the knowledge base. The domains requiring fuzzy reasoning would select a meta-expert system from EMYCIN, Meta-DENDRAL, KAS, or EXPERT (Weiss and Kulikowski 1979); the need for exact reasoning would direct us to a rule based programming language OPS5 (Forgy 1981), HEARSAY-III, ROSIE (Fain et al 1981, 1882), or a frame based system RLL (based on a cognitive model of human thought - Greiner and Lenat 1980). Having these tools facilitates a quick and easy expert system design.

This paper describes typical expert system architecture and facilities. Many other features characteristic of individual systems can be found in the references.

Summary

The paper reviews various aspects of expert and knowledge based systems. It demonstrates the superiority of these over the traditional computer systems. There are, however, many unresolved problems with the new systems, e.g. the unmanageable complexity of their architecture; difficulties for knowledge engineers in system maintenance; etc.
Further research could resolve the whole range of problems associated with the new systems by the development of appropriate tools. Considering the attractive, some would say revolutionary, facilities offered by the new generation of computer systems, such research could be potentially highly fruitful to both the computer industry and academic centres in Australia.

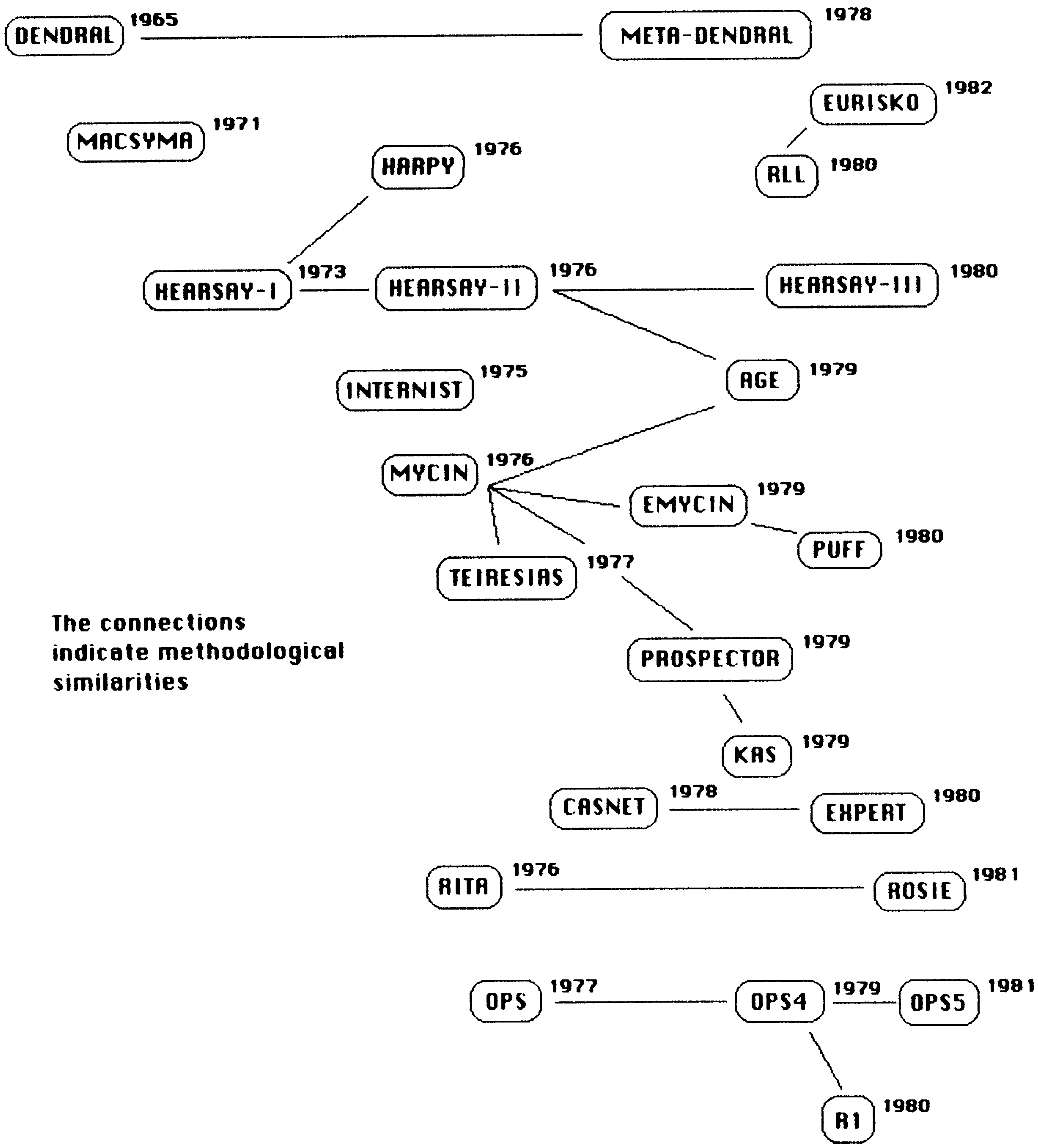


Figure 10 - Development of more important ESs

APPENDIX - A summary of expert and knowledge based systems

Expert System	Bibliographic Reference
AGE	Nii and Aiello 1979
AL/X	Reiter 1980
AQ11	Michalski and Chilausky 1980
CADUCEUS	Pople, Myers and Miller 1975
CASNET	Weiss, Kulikowski and Safir 1977
CBC	Hart 1975
CENTAUR	Aikins 1980
CONGEN	Carhart and Smith 1976
	Smith and Carhart 1978
CRYSALIS	Engelmore and Terry 1979
	Engelmore and Nii 1977
DART	Bennet and Hollander 1981
DENDRAL	Lindsay et al 1980
DTA	Gorry, Silverman and Pauker 1978
	Swarout 1977
EMYCIN	Van Melle 1980
EURISKO	Lenat, Sutherland and Gibbons 1982
EXPERT	Weiss and Kulikowski 1979
GUIDON	Clancey 1979
HEADMED	Heiser, Brooks and Ballard 1978
HEARSAY-II	Erman et al 1980
HEARSAY-III	Balzer et al 1980
HODGKINS	Safrans et al 1976
INTERNIST	Pople 1977
IRIS	Trigoboff and Kulikowski 1977
KAS	Hayes-Roth, Waterman, Lenat 1983
MACSYMA	Martin and Fateman 1971
Meta-DENDRAL	Buchanan and Feigenbaum 1978
MOLGEN	Stefik 1980
MYCIN	Shortliffe 1976
NEOMYCIN	Clancey and Letsinger 1981
NLS-SCHOLAR	Grignetti, Hausman, and Gould 1975
ONCOCIN	Shortliffe et al 1981

Expert System	Bibliographic Reference
OPS	Forgy and McDermot 1977
PIP	Pauker et al 1976
	Szolovits and Pauker 1978
PROSPECTOR	Duda, Gashing and Hart 1979
	Duda et al 1978
PUFF	Kunz et al 1978
RITA	Anderson and Gilloglie 1976
ROSIE	Fain et al 1981, 1982
RX	Blum and Wiederhold 1978
R1	McDermot 1981
SACON	Bennet et al 1978
SCHOLAR	Carbonell 1970
SECS	Wipke et al 1977
SOPHIE	Brown and Burton 1978
SPERIL	Ishizuka et al 1981
SYNCHEM	Gelernter et al 1977
TEIRESIAS	Davis and Lenat 1982
TIMM	Kornell 1984
	Hunt 1984
VM	Fagan 1979
WHY	Stevens, Collins and Goldin 1978

Also a range of new products available for micro-computers

APES	IBM PC, 68000	Logic Based Systems (UK)
ES/P Advisor	not specified	Expert Systems Int (USA)
ExpertEase	IBM PC	Intelligent Terminals (UK)
Lisp	various	Variety of implementations
Micro-Expert	IBM PC	Isis Systems (UK)
Micro-Prolog	IBM PC	Logic Programming Assc (UK)
Siclare Prolog	Sinclare Spectrum	

others

Bibliography

Aikins, J. (1980). "Prototypes and production rules: A knowledge representation for computer consultation," Research Report STAN-CSD-80-814, Computer Science Department, Stanford University.

Anderson, R.H., and Gillogly, J.J. (1976). "The RAND intelligent terminal (RITA) as a network design access aid," Proceedings of the AFIPS National Computer Conference, 501-509.

Balzer, R., Erman, L.D., London, P., and Williams, C. (1980). "HEARSAY-III: A domain-independent framework for expert systems," National Conference of the American Association for Artificial Intelligence 1, 108-110.

Barnet, J.A. (1981). "Computational methods for mathematical theory of evidence," International Joint Conference on Artificial Intelligence 7, 868-875.

Bennet, J.S., Creary, L.A., Englemore, R.M., and Melosh, R.E. (1978). "SACON: A knowledge-based consultant in structural analysis," Research Report HPP-78-23, Heuristic Programming Project, Computer Science Department, Stanford University, California.

Bennet, J.S. and Hollander, C.R. (1981). "DART: an expert system for computer fault diagnosis," International Joint Conference on Artificial Intelligence 81, 843-845.

Blum, R.L., and Wiederhold, G. (1978). "Infering knowledge from clinical data banks: Utilizing techniques from artificial intelligence," Proceedings of the Second Annual Symposium on Computer Application in Medical Care, IEEE, Washington, D.C., 303-307.

Brown, J.S., and Burton, R.R. (1978). "Multiple representation of knowledge for tutorial reasoning." In Bobrow, D.G., and Collind, A. (eds) Representation and Understanding: Studies in Cognitive Science, New York: Academic Press, 311-349.

Buchanan, B.G., and Feigenbaum, E.A. (1978). "DENDRAL and Meta-DENDRAL: their applications dimension," Journal of Artificial Intelligence 11:5-24.

Burstall, R.M., Collins, J.S., and Popplestone, R.J. (1971). Programming in POP-2. Edinburgh University Press.

Carbonell, J.R. (1970). "AI in CAI: An artificial intelligence approach to computer-aided instruction," IEEE Transactions on Man-Machine Systems MMS-11(4):190-202.

Carhart, R.E. and Smith, D.H. (1976). "Applications of artificial intelligence for chemical inference XX. Intelligent use of constraints in computer assisted structure elucidation," Computers and Chemistry 1, 79.

Clancey, W.J. (1979). "Transfer of rule-based expertise through a tutorial dialogue." Research Report STAN-CS-769, Computer Science Department, Stanford University.

Clancey, W.J. (1981). "The epistemology of a rule-based expert system: A framework for understanding," Research Report STAN-CS-81-896, Computer Science Department, Stanford University.

Clancey, W.J., and Letsinger, R. (1981). "NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching," International Joint Conference on Artificial Intelligence 7, 829-836.

Clocksin, W.F., and Mellish, C.S. (1981). Programming in Prolog, Springer-Verlag.

Davis, D.B. (1984). "English: The newest computer language," High Technology, vol 4, 2:59-64.

Davis, R., and Lenat, D.B. (1982). Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill, 229-490.

Duda, R., Gashing, J., Hart, P.E. (1979). "Model design in the PROSPECTOR consultant system for mineral exploration." In Michie, D. (ed), Expert Systems in the Micro-Electronic Age, Edinburgh University Press, 153-167.

Duda, R., Gashing, J., Hart, P.E., Conolige, K., Reboh, R., Barret, P., and Slocum, J. (1978). "Development of the PROSPECTOR consultation system for mineral explorations," Final Report, SRI Projects 5821, 6415, Stanford Research Institute International, Inc., Menlo Park, California.

Englemore, R.E. and Nii, H.P. (1977). "A knowledge-based system for the interpretation of protein x-ray crystallographic data." Heuristic Programming Project Report HPP-77-2, Computer Science Department, Stanford University.

Englemore, R.E. and Terry, A. (1979). "Structure and fundation of CRYSALIS system," International Joint Conference on Artificial Intelligence 6, 250-256.

Erman, L.D., Hayes-Roth, F., Lesser, V., and Reddy, D. (1980). "The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty," Computing Surveys 12, 2: 213-253.

Ernst, G., and Newell, A. (1969). GPS: A Case Study in Generality and Problem Solving, Academic Press.

Fagan, L. (1979). "Knowledge engineering for dynamic clinical settings: Giving advise in the intensive care unit," Doctoral Dissertation, Computer Science Department, Stanford University.

Fain, J., Gorlin, D., Hayes-Roth, F., Rosenschein, S.J., Sowizral, H., and Waterman, D. (1981). "The ROSIE language reference manual," Technical Report N-1647-ARPA, Rand Corp., Santa Monica, California.

Fain, J., Hayes-Roth, F., Sowizral, H., and Waterman, D. (1982). "Programming in ROSIE: An introduction by means of examples," Technical Report N-1646-ARPA, Rand Corp., Santa Monica, California.

Feldman, J.A., Low, J.R., Swinehart, D.C., and Taylor, R.H. (1972). "Recent developments in SAIL," Research Report STAN-CS-308, Computer Science Department, Stanford University.

Forgy, C.L. (1981) "The OPS5 user's manual," Technical Report CMU-CS-81-135, Computer Science Department, Carnegie Mellon University.

Forgy, C.L. and McDermot, J. (1977). "OPS, a domain independent production system language," International Joint Conference on Artificial Intelligence 5, 933-939.

Gelernter, H.L., Sanders, A.F., Larsen, D.L., Agarival, K.K., Boivie, R.H., Sprintzer, G.A., and Searleman, J.E. (1977). "Empirical explorations of SYNCHEM," Science 197:1041-1049.

Goldstine, H. (1972). The Computer from Pascal to von Neumann. Princeton, N.J. Princeton University Press.

Gorry, G.A., Silverman, H., and Pauker, S.G. (1978). "Capturing clinical expertise: A computer program that considers clinical response to digitalis," American Journal of Medicine 64, 452-460.

Green, C. (1969). "The application of theorem proving to question answering systems," Memo AIM-96, STAN-CS-69-178, Computer Science Department, Stanford University.

Greiner, R., and Lenat, D. (1980). "A representation language language," National Conference of the American Association for Artificial Intelligence 1, 165-169.

Grignetti, M.C., Hausmann, C., and Gould,L. (1975). "An intelligent on-line assistant and tutor - NLS-SCHOLAR," Proceedings of the National Computer Conference, San Diego, California, 775-781.

Hart, P.E. (1975). "Progress on a computer based consultant," International Joint Conference on Artificial Intelligence 4, 831-841.

Hart, P.E., and Duda, R.O. (1977). "PROSPECTOR-- A computer based consultation system for mineral explorations," Technical Note 155, AI Centre, SRI International.

Hayes-Roth, F., Waterman, D.A., and Lenat, D.B. (1983). Building Expert Systems, Addison-Wesley.

Heiser, J.F., Brooks, R.E., and Ballard, J.P. (1978). "Progress report: A computerised psychopharmacology advisor," Proceedings of the Eleventh Collegium Internationale Neuro-Psychopharmacologicum, Vienna, Austria.

Hewitt, C. (1971). "Description and theoretical analysis (using schemas) of PLANNER: A language for proving theorems and manipulating models in a robot," MIT Laboratory.

Hunt, A. (1984). "The development of an expert system that assesses the vulnerability of assets to subnational adversaries," General Research Corporation.

Ishizuka, M., Fu, K.S., and Yao, J.T.P. (1981). "Inexact reasoning for rule-based damage assesment of existing structures," International Joint Conference on Artificial Intelligence 81, 837-842.

Kornell, J. (1984). "Automated knowledge acquisition and representation," General Research Corporation.

Kowalski, R. (1979). Logic for Problem Solving, North-Holland.

Kunz, J.C., Fallat, R.J., McClung, D.H., Osborn, J.J., Votteri, R.A., Nii, H.P., Aikins, J.S., Fagan, L.M., and Feigenbaum, E.A. (1978). "A physiological rule-based systems for interpreting pulmonary function test results," Research Report HPP-78-19, Heuristic Programming Project, Computer Science Department, Stanford University.

Le Faivre, R.A. (1977). Fuzzy Reference Manual, Computer Science Department, Rutgers University.

Lenat, D.B., Sutherland, W.R., and Gibbons, J. (1982). "Heuristic search for new microcircuit structures: An application of artificial intelligence," Artificial Intelligence Magazine 3, 17-33.

Lindsay, R.K., Buchanan, B.G., and Feigenbaum, E.A., and Lederberg, J. (1980). Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project, McGraw-Hill.

Mamdani, E.H. and Gaines, B.R. (1981). Fuzzy Reasoning and its Application. Academic Press.

Martin, W.A. and Fateman, R.J. (1971). "The MACSYMA system," Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, 55-75.

McCarthy, J. (1961). "Recursive functions of symbolic expression and computation by machines," Communication of ACM 4:185-195.

McCarthy, J., Abrahams, P.W., Edwards, D.J., Hart, T.P., and Levin, M.I. (1962). Lisp 1.5 Programmers Manual, MIT Press.

McDermot, D.V. (1981). "R1: The formative years," Artificial Intelligence Magazine, 2:21-29.

Michalski, R.S. and Chilausky, R.L. (1980). "Knowledge acquisition by encoding expert rules versus computer induction from examples," International Journal of Man-Machine Studies 12, 63-87.

Newell, A., and Simon, H.A. (1963). "GPS: A program that simulates human thought," in Feigenbaum, E.A. and Feldman, J.A. (eds) Computers and Thought, NY: McGraw-Hill.

Nii, H.P. and Aiello, N. (1979). "AGE (Attempt to Generalize): A knowledge-based program for building knowledge-based programs," International Joint Conference on Artificial Intelligence 6, 645-655.

Pauker, S., Gorry, G.A., Kaisserer, J., and Schwartz, W. (1976). "Towards the simulation of clinical cognition - Taking the present illness by computer," American Journal of Medicine 60:981-996.

Pople, H.E. (1977). "The formation of composite hypothesis in diagnostic problem solving - An exercice in synthetic reasoning," International Joint Conference on Artitificial Intelligence 5, 1030-1037.

Pople, H.E., Myers, Jr.J.D., and Miller, R.A. (1975). "DIALOG: A model of diagnostic logic for internal medicine," International Joint Conference on Artificial Intelligence 4, 848-855.

Reiter, J. (1980). "AL/X: An expert system using plausible inference," Intelligent Terminals Ltd, Oxford.

Robinson, J.A. (1979). Logic, Form and Function: The Mechanization of Deductive Reasoning, Edinburgh University Press.

Robinson, J.A. (1983). "Logic Programming: Past, Present and Future," ICOT Technical Report 015.

Ruliffson, J.F., Waldinger, J.A., and Derkson, J.A. (1971). "A language for writing problem-solving programs." IFIPS Congress 71.

Safrans, C., Desforges, J., and Tsichlis, P. (1976). "Diagnostic planning and cancer management," Research Report TR-169, Laboratory for Computer Science, Massachussets Institute of Technology.

Shortliffe, E.H. (1976). Computer-Based Medical Consultations: MYCIN, New York: American Elsevier.

Shortliffe, E.H., Scott, A.C., Bischoff, M.B., Campbell, A.B., Van Melle, W., and Jacobs, C.D. (1981). "ONCOCIN: An expert system for oncology protocol management," International Joint Conference on Artificial Intelligence 7.

Siklossy, L. (1976). Let's Talk LISP, Prentice-Hall.

Simon, H.A. (1969). The Science of the Artificial, The MIT Press.

Smith, D.H. and Carhart, R.E. (1978). "Structure elucidation based on computer analysis of high and low resolution mass spectral data." In Gross, M.L. (ed) , High Performance Mass Spectrometry: Chemical Applications, Washington, D.C.: American Chemical Society, 325.

Stefik, M. (1980). "Planning with constraints," Research Report STAN-CS-80-784, Computer Science Department, Stanford University.

Stevens, A.L., Collins, A., and Goldin, S. (1978). "Diagnosing student's misconceptions in causal models," BBN Research Report 3786, Bolt Beranek and Newman, Inc., Cambridge, Mass.

Sussman, G.J., and McDermot, D.V. (1972). "Why conniving is better than planning," AI Memo 255A, AI Laboratory, MIT.

Swartout, W. (1977). "A digitalis therapy advisor with explanations," International Joint Conference on Artificial Intelligence 5, 819-825.

Szolovits, P., and Pauker, S. (1978). "Categorical and probabilistic resoning in medical diagnosis," Arificial Intelligence Journal 11:115-154.

Trigoboff, M., and Kulikowski, C. (1977). "IRIS: A system for the propagation of inferences in a semantic net," International Joint Conference on Artificial Intelligence 5, 274-280.

Van Melle, W. (1980). "A domain independent system that aids in constructing consultation programs," Research Report STAN-CS-80-820, Computer Science Department, Stanford University.

Weiss, S.M., and Kulikowski, C. (1979). "EXPERT: A system for developing consultation nodes," International Joint Conference on Artificial Intelligence 6, 942-947.

Weiss, S.M., Kulikowski, C., and Safir, A. (1977). "A model-based consultation system for the long-term management of glaucoma," International Joint Conference on Artificial Intelligence 5, 826-832.

Wipke, W.T., Braun, H., Smith, G., Choplin, F., and Sieber, W. (1977). "SECS - simulation and evaluation of chemical synthesis: Strategy and planning." In Wipke, W.T. and House, W.J. (eds) Computer Assisted Organic Synthesis, Washington, D.C.: American Chemical Society, 97-127.

Zadeh, L.A. (1977). "PRUF - a meaning representation language for natural languages," in Mamdani, E.H. and Gaines, B.R. (eds). Fuzzy Reasoning and its Application, Academic Press, 1983, 1-66.