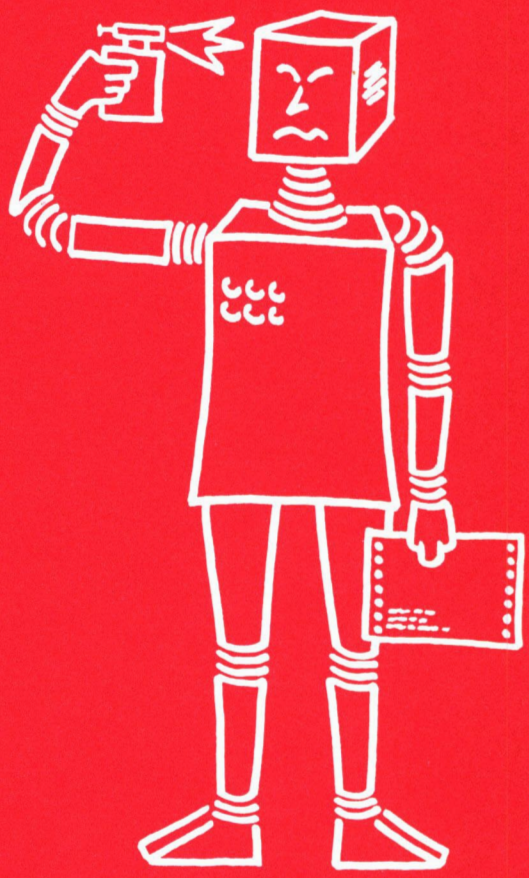
SEKI-PROJEKT

SEKI MEMO

Institut für Informatik III
Universität Bonn
Bertha-von-Suttner-Platz 6
D 5300 Bonn 1, W. Germany

Institut für Informatik I
Universität Karlsruhe
Postfach 6380
D-7500 Karlsruhe 1, W. Germany



MEMO SEKI-BN-81-03

EXPERT SYSTEMS:
STATE OF THE ART AND FUTURE PROSPECTS

Peter Raulefs

Expert Systems:
State of the Art and Future Prospects

Peter Raulefs

SEKI-Projekt
Institut fur Informatik III
Universitat Bonn
Postfach 1220
D-5300 Bonn 1, West Germany

1. Introduction

Knowledge-based systems, and expert systems in particular, have recently gained significant economic and scientific importance because of

- a rapidly increasing demand for expert consultancy.
- tremendous cost reductions following the mechanization of expertise otherwise only available from highly trained specialists.
- accomplishments of advanced expert systems.

Furthermore, it becomes increasingly apparent that knowledge-based systems constitute an evolving tool for mechanizing fields in science and engineering that primarily rely on judgmental, experiental, and heuristic knowledge, and have so far been repugnant with mechanization, yet experienced slow growth in comparison with other fields.

This paper is intended to give a brief and introductury survey on the present state of knowledge engineering and future trends. In its present state of infancy, knowledge engineering withholds textbook expositions. The field consists of a growing number of expert systems. Apart from pattern-directed representation and search, there are hardly any general techniques for constructing expert systems. A thorough presentation of the field would therefore require a detailed and comparative analysis of various expert systems. As this is far beyond the scope of this paper, we merely try to work out some basic and unifying issues: Section 2 analyzes objectives for developing expert systems, followed by a sample of expert systems in Section 3. In Section 4, an attempt is made to work out some basic design principles for expert systems, as well as criteria for evaluating them. Summarizing conclusions are compiled in Section 5.

Notation. XP stands for (human) expert(s).
XPS stands for expert system(s).

2. Objectives of Expert Systems

There are primarily two objectives for developing XPS:

- To mechanize the activities of human experts.
- To have representations of fields of science amenable to mechanical manipulation.

This section analyzes these objectives and establishes criteria for determining the demand for XPS.

2.1. Consultancy Systems

XPS are intended to at least partially mechanize the activities of a (human)XP. The objectives for designing an XPS therefore derive from how we caracterize a human XP.

[XP] A (human) XP is a specialist for a distinguished **area of expertise**; and is **competent in consulting** a client by applying facts and techniques of the area of expertise. An XP is therefore characterized by

consulting competence consisting of the abilities to

- **understand** a client's problems which often arise outside the area of expertise of the XP.
  recognize from the queries of a client where applying own expertise may contribute to solve the client's problems.
- **transform** a client's problems into a model of the own area of expertise.
- **solve** a client's problems.
- **communicate** solutions to a client.
- **explain** the reasoning resulting in a solution to the client.
- help the client to apply and **integrate** solutions.

> An XPS is a computer system mechanizing to some extent the activities of an XP. A **consultancy system** consists of a client conversing with an XP, with the XP aided by an XPS. There is a great variety in the extent to which an XPS may mechanize the activities of an XP. Total mechanization of an XP by an XPS requires mechanizing both expertise (i.e. knowledge and skills) and consulting competence. In most applications, however, it will only be possible or cost-effective to partially mechanize expertise and consulting competence.

The extent to which mechanizing consulting competence is desirable very much depends on the client. A client who is quite familiar with the area of expertise of an XP/XPS may ask only for a minimum of consulting competence, in contrast to a client who doesn't even know in which way knowledge and skills in the area of expertise may affect his own problems.

Mechanization introduces an additional difficulty: Mechanical systems operate on machine-oriented representations. Clients converse in natural or somewhat formalized technical language. To achieve the abilities comprising consulting competence, an XPS must therefore have the following capabilities:

- **Language Comprehension Capability: Understand** client's queries, **transform** them into internal representation, and **recognize** approaches to apply own expertise.
- **Solution Capability:** Solve client's problems where own expertise applies.
- **Answer Construction Capability: Construct answer** which explains solution to the client so that the client can understand and apply it.
- **Solution Explanation Capabiltity: Explain reasoning** resulting in the answer to the client so that the client may judge the reliability of the solution given by the XPS.
- **Solution Application Capability:** Help client in **applying and integrating** solution.

[Summary]. The activities of an XP aided or mechanized by an XPS cannot be viewed in isolation. Instead, an XP/XPS together with a client form a **consultancy system**, where they cooperate in finding, evaluating, and applying solutions to problems arising from the client's queries.

2.2. Determining the Demand for Expert Systems

To determine demand and necessity for developing XPS, we consider three criteria:

[**Economic Criterion**]	The cost-gains relationship.
[**Scientific Criterion I**]	Amount of new insights and and techniques for mechanizing intellectual activities (**contributions to A I**).
[**Scientific Criterion II**]	Benefits of having representations of fields of science.

This paper will not pursue the [Economic Criterion], but concentrates on the [Scientific Criteria]. In particular, justifying [Scientific Criterion II] requires further explanation. We will argue that the most substantial contributions of XPS will consist in XPS substituting textbook expositions for both exposing fields of science, and for being instrumental in their application and further development. We start from four observations on the evolution of a particular variety of fields of science. Our conclusions justifying [Scientific Criterion II] will be based on these observations.

[**Observation 1**] In the history of science, those fields have developed most rapidly which allow the formation of theories to represent the body of knowledge of such a field. Theory formation is the most effective way of making knowledge both available and applicable ("Am praktischsten ist eine gute Theorie" - A.Einstein).

[**Observation 2**] There are fields which are adverse to theory formation, as

- they comprise a large amount of isolated chunks of knowledge which are difficult to structure and cannot be abstracted s.t.

they occur as instances of a few general principles.
- they typically employ heuristic techniques for reasoning and search.

We call such a field a **diffuse field** of science in contrast to **well-structured fields** allowing extensive theory formation.

[**Observation 3**] As knowledge of diffuse fields cannot be condensed in theories, competence in a diffuse field cannot be acquired by short and systematic training, but it is typically obtained after extensive professional activity ("gathering experience").

[**Observation 4**] Many supposedly well-structured fields have diffuse sub-fields.

Examples: (1) Number theory is supposedly well-structured. But the sub-field of expertise about how to find proofs in number theory is diffuse.

(2) The chemistry of large organic molecules is supposedly well-structured. But the sub-field of expertise on how to determine the structure of large organic molecules is diffuse.

From these observations, we conclude the following facts:
- Most fields of science contain diffuse sub-fields.
- Diffuse fields cannot be condensed in theories.
- Acquiring expertise in a diffuse field is extremely expensive.

It is difficult to judge the reliability of expertise on a diffuse field.

The development of diffuse fields is slow compared to the growth of well-structured fields.

Developing applications of the expertise of a diffuse field is much more difficult than applying knowledge and skills belonging to a well-structured field.

Human XP require extremely expensive training and perform worst on diffuse fields.

There are three main consequences for the demand for XPS:

I. XPS should be developed for providing expertise and consulting competence on diffuse fields.

II. XPS constitute instruments to further develop and apply diffuse fields.

III. The benefits of XPS increase with the accumulation of expertise which is both rapidly accessible and rapidly applicable.

3. Accomplishments of Expert Systems

To give an impression what can be achieved by XPS, we give a brief survey of successful XPS. This survey is by no means intended to be exhaustive.

Fields of Competence	Expert System	Area of Expertise	References
Medicin	MYCIN	bacteria identification & antibiotics therapy	[SHO 76]
	Digitalis Therapy Advisor		[SWA 77]
	PUFF	lung test interpretation	[KUN 78]
	INTERNIST	internal medicine diagnosis	[PMM 77]
	VM	iron-lung control	[FAG 78]
Chemistry	DENDRAL	identification of chemical compounds	[BF 78] [STE 78]
	CRYSALIS	structure of protein molecules	[ET 79]
	MOLGEN	molecular genetics	[FRI 79]
	SECS	design of organic synthesis	[WIP 74]
Mechanics	MECHANO	solving mechanics problems	[BUN 78]
	SACON	structural analysis consultant (for bridges, houses,etc.)	[BCEM 78]
Geology	PROSPECTOR	mineral prospecting oil prospecting	[HDE 78]
Plant diseases	Diagnosis of plant diseases		[MC 79]
Electric Circuits	EL	electric circuit analysis	[SS 77]
Programming	PECOS	automatic programming	[BAR 79]
	APE		[BOR 81]

4. Design Principles for Expert Systems

As it turns out in Sect.4.1, it is hardly avoidable to design an XPS as a **pattern directed inference system (PDIS).** Therefore, this Section is actually a survey on how to design a PDIS. Very often, the core of a PDIS is a **production system.** Production systems are described in Sect.4.2, while Sect.4.3 mentions alternative mechanisms for implementing PDIS. Finally, Sect.4.4 develops criteria for evaluating the design of PDIS.

4.1. Basic Considerations

By our analysis of Sect.3, there are three requirements to consider when designing an XPS:

[RQ 1] XPS are intended to manipulate vast quantities of poorly structured knowledge and skills. Hence, XPS require a representation of knowledge and skills (=expertise) which supports
- rapid detection of expertise from situations where the expertise is immediately applicable towards achieving goals.
- the acquisition of large amounts of new knowledge.

[RQ 2] XPS should incorporate all domain-specific reasoning mechanisms and problem solving skills belonging to the respective area of expertise.

[RQ 3] XPS should have capabilities for comprehending client's queries, and for constructing, explaining, and applying solutions.

In this paper, we concentrate on design principles intended to meet requirements [RQ 1,2].

Abstraction is the most effective method for representing large amounts of knowledge. The two most important abstraction techniques are **schematization** and **axiomatization.**

Schematization. A schema, or **pattern,** is a syntactic object to which applying a substitution results in an instance of the pattern. Hence, a pattern is a description of the set of all of its instances, called the **extension** of the pattern. In other words, **a pattern is an abstraction of its extension.**

To illustrate the significance of abstraction achieved by patterns we quote an example from [MIC 80] about chess end games. Decisions for sequences of winning moves are based on **diagnostic patterns** characterizing typical situations on the chess board. This table compares the search space for small sub-domains with the number of diagnostic patterns which suffice to decide on winning moves.

Situation Patterns	Search Space	No. of Characterizing
King, Rook vs. King	40 000	10
King, Pawn vs. King	100 000	20
King, Knight vs. King, Rook	2 000 000	30

Axiomatization. An area of knowledge is axiomatized by a **formal system** consisting of
- **axioms**, i.e. chunks of knowledge considered to be primitive.
- **deduction rules**, i.e. rules specifying how chunks of knowledge may be derived from each other.
- **meta-rules**, i.e. rules specifying how deduction rules may be applied, or altered and applied to achieve deductions in a goal-oriented way.

Pattern-directed inference systems (PDIS) integrate schematization and axiomatization, and form the core constituent of almost any XPS.

4.2. Production Systems

Production systems are the most commonly occurring constituents of PDIS. We give a short and condensed survey on design principles of production systems.

4.2.1. Architecture. A production system consists of
- a **data base**, i.e. a system of syntactically uniform encodings **(data)** of chunks of knowledge.
- a **production base**, i.e. a system of **production rules**; a production rule is a pair <situation> -> <action>, where <situation> is a pattern, and the effect of applying a production rule is determined by the interpreter of the production system.
- an **interpreter** consisting of
  - a **pattern matcher** matching data of the data base to <situation>-patterns of productions in the production base.
  - an **executor** for executing <action>s, possibly changing data and production base.
  - a **control** to select production rules to be considered by the pattern matcher and executor.

4.2.2. Recognize-Act-Cycle. A production system operates according to a recognize-act-cycle:

(1) **recognize**: determine a production rule from the production base which is **applicable**, i.e. the state of the data base is in the extension described by the

<situation>-part of the production rule. The outcome of the recognize-phase consists of
- an indication of an applicable production rule;
- possibly bindings to match variables occurring in both <situation>- and <action>-part of the production rule.

(2) **act:** execute the <action>-part of the production rule determined in the recognize-phase, using bindings to match variables established in that phase.

The recognize-phase is carried out by control and pattern matcher, the act-phase by the executor of the interpreter.

Often there are several applicable production rules. If the recognize-phase proposes several applicable production rules, the interpreter control has to decide which rule is to be executed (see 4.2.5)

4.2.3. Direction of derivations. Production systems may work in **forward** or **backward direction.**

In the forward direction, production rules are applied whenever the <situation>-part is satisfied, regardless whether effects of the <action>s are desirable or not. The danger of this **forward reasoning strategy** is that production rules may generate new data added to the data base, although such data are irrelevant to reach an intended goal state.

Backward reasoning first considers a goal situation of the data base, and determines which production rule(s) appear(s) to be most suitable to approach this goal given the current state. Having decided on such a rule, applying this rule now requires that the data base satisfies the <situation>-part of the rule. If this is not so, we obtain a **subgoal** and backward reasoning again is applied to reach this subgoal. Upon having reached an applicable production rule, the reverse of the sequence of rules that have been selected so far constitutes a **plan** s.t. executing the plan achieves the intended goal.

Backward reasoning has the advantage of being **goal-oriented,** i.e. it avoids littering the data base with irrelevant data that forward reasoning might produce. Another advantage of backward reasoning is that it is a strategy to avoid applying failing sequences of production rules in derivations with data base states where several production rules apply. A disadvantage of backward reasoning is that it constitutes a rather inflexible control mechanism. In practise, one often chooses a **mixed strategy** using both forward and backward reasoning.

4.2.4. Structure of the Data Base. The simplest way of constructing a data base consists of assembling a collection of encodings of chunks of knowledge. Such a collection may be implemented in a standard way, e.g. providing quick access using hash coded keys. However, structuring a data base may itself be part of knowledge representation. As an example, data items may be stored

at nodes of trees s.t. access to data requires traversing the tree along paths starting at the root; the ancestor relationship of data items may then be used for representing semantic properties.

4.2.5. Control. The control part of the interpreter, i.e. the mechanism for controlling the application of production rules, is of crucial influence on the performance of a production system. There are two basic problems to solve when designing the control of an interpreter:

- **selection** of production rules to be processed by the recognition-act-cycle.
- **conflict resolution**, i.e. deciding which rule is to be applied after several rules have been recognized to be applicable.

4.2.5.1. Selection mechanisms. The main objective of selecting production rules for processing by the recognize-act-cycle is to avoid the inspection of inapplicable production rules. The rule selection phase may present just one, or a possibly structured collection of production rules to the recognize-act-cycle.

A coarse way of avoiding unwanted inspection of production rules consists in **decomposing the production base into several production bases**, and control access to a production base as a function of the current state of the production system. This technique often arises from decomposing the area of expertise s.t. each production base becomes a **domain expert.** If the effects of different domain experts are either independent of each other, or are reasonably coordinated, putting interpreters on different processors results in a system of **communicating production systems.**

If in a single production base several production rules are applicable, If in a single production base several production rules are applicable, there are three basic strategies to proceed:

(1) **First-encounter-strategy:** Select the production rule which happens to be the first being encountered when testing for applicability. For this strategy, production rules are often implemented as a lifo-stack {resp. lifo-queue} s.t. the most recently applied production rules will be inspected first {last} **(attention focussing)**
(2) **Conflict-resolution:** Collect all applicable production rules and employ some explicit conflict-resolution technique for making a choice.
(3) **Try all:** Execute all applicable production rules, resulting in several states which need to be worked on independently. Clearly, this srategy can only be pursued if a combinatorial explosion of paths can be avoided.

A production system **terminates** iff no production rule is applicable. Important properties of production systems ar **finite and unit termination.** If production rules are **term rewriting rules,** there are formal methods to check for these properties [HO 80].

An important species of production systems are those employing

fuzzy reasoning. Here, data items are tagged with a "certainty factor", and producdtion rules compute such certainty factors for all new data they produce. The <situation>-patterns of production rules will contain match variables qualified with requirements on certainty factors of the data to be matched. Executing different sequences of production rules all starting from the same state of the of the production system may result in asserting different data, each of them to some degree of probability. Deriving a datum by several sequences of production rules may be taken to be an indication for increased evidence, while deriving conflicting data may be viewed as diminishing the evidence of such data. From these considerations we see that for fuzzy reasoning, the Try-all-section-strategy goes along with a mechanism of **evidence amplification/diminuation.** An example of such a system is MYCIN [SHO 76]; other mechanisms for evidence amplification/diminuation are investigated in [WAH 80].

4.2.5.2. Conflict resolution techniques. If several applicable production rules have been determined, and the Try-all-strategy is not employed, a decision for selecting one of the conflicting rules must be made. One typically collects all such production rules in an **agenda**, and applies a seperate **conflict resolution mechanism** to the agenda, resulting in a rule to be executed.

There are three classes of conflict resolution techniques:

(1) **Rule-ordering.** This technique assigns each rule a **priority value** s.t. applicable rules become linearly ordered w.r.t. priority value. Quite often, the following priority measures are applied:
   - generality order:
     more "specific" rules are assigned a higher priority; here, a rule <sit 1> -> <act 1> is more specific than rule <sit 2> -> <act 2> iff (under forward reasoning) the extension of <sit 1> {backward reasoning:<act 1>} is included in the extension of <sit 2> {<act 2>}.
   - recency order:
     rule R1 has a higher priority than rule R2 iff R1 has been applied more recently (corresponds to attention focussing).

(2) **Data-controlled conflict resolution.** Priority values are attached to items of the data base. A rule of the agenda is selected iff the sum of all priority values of matching data items is maximal.

(3) **Meta-rules.** Meta-rules operating on the production rules of the agenda decide which production rule is to be executed next.

4.3. Alternative Techniques for Realizing Pattern Directed Inference Systems

Because of its central importance, different techniques for realizing pattern directed inference have emerged independently

from various sub-fields of AI. The following is a list of some such techniques:

(1) **Horn-clause deductions.** A Horn clause L <= L1 & L2 & ...& Ln consists of literals (atomic predicate calculus formulas) forming an implication with a conjunction of literals being the premise, and a single literal being the conclusions. In disjunctive form, the above Horn clause is -L **or** L1 **or** Ln. Such a Horn clause can be interpreted as a production rule with premise and conclusion [WAR 77] employ resolution theorem-provers to interpret Horn clauses with backward reasoning.

(2) **PLANNER consequent/antecedent theorems** [HEW 72] are procedural implementations of predicate calculus implications interpreted as production rules.

(3) **Augmented Transition Networks** [WOO 70] consisting of states as nodes, and having production rules attached to the edges can be utilized as a control mechanism s.t. traversing an ATN consists of executing state transforming production rules attached to the edges being traversed.

(4) **Agents** [FRV 81] provide a way of realizing systems of communicating concurrent production systems.

4.4. Criteria for Evaluating Pattern Directed Inference Systems

Which mechanisms support the development of PDIS at minimal cost, yet achieving maximal performance? This section discusses such mechanisms and criteria to evaluate them.

4.4.1. Constructive criteria. High performance while minimizing development costs is primarily achieved by adhering to the following constructive criteria.

- **Performance.**
  - (1) **Completeness** of results produced by a PDIS depends on
    - (1.1) the completeness of the knowledge base. To achieve this is a problem of **knowledge acquisition** and **knowledge base validation.**
    - (1.2) the completeness of the deductive machinery, e.g. production rules and control of a production system. Techniques for achieving this so far only exist for logical calculi and term-rewriting systems.
  - (2) **Correctness** of results produced by PDIS
    - (2.1) depends on the correctness of the knowledge base, again a problem of **knowledge acquisition** and **knowledge validation.**
    - (2.2) depends on the correctness of the deductive machinery (see comment under (1.2)).
    - (2.3) is supported by the **transparency** of the I/O-behavior of a PDIS from inspecting its knowledge base and deductive machinery.
  - (3) **Efficiency** is especially affected by
    - (3.1) efficient pattern matching.
    - (3.2) **knowledge organization** in the data base s.t. unsuccessful search is excluded as much as possible.

(3.3) the quality of rules and control, again a problem of **knowledge acquisition.**

¤ **Development costs.**
(1) **Knowledge acquisition** is the overwhelming fraction in the development of a PDIS.
(2) **Modularity** is the degree of separation of functional units, s.t. changes, deletions, and additions do not entail modifications of other functional units. As system development proceeds incrementally, lack of modularity greatly increases development costs.
These constructive criteria imply the technical criteria discussed next.

4.4.2. Technical criteria. The above constructive criteria are met if the following technical criteria are satisfied: modularity, transparency, and support of knowledge acquisition.

¤ **Modularity.** The following observations provide guidelines for achieving modularity in production systems:
(1) Modularity is increased the more chunks of knowledge are represented in single production rules resp. data items. The amount of knowledge represented in such chunks is often referred to as **granularity.** There is often a trade-off between granularity and effiency.
(2) Backward reasoning supports modularity better than forward reasoning. This is due to the fact that conflict resolution can be better modularized for backward reasoning.
(3) Modularity decreases the more control mechanisms are coded into production rules. It is preferable to seperate control as e.g. in meta-rules, ATNs, agent-systems.

¤ **Transparency.** The transparency of production systems is usually low. The reason is that transparent I/O-behavior is achieved by stamping control structures on state changing actions (as higher programming languages do). Again, a cure for this deficiency of production system consists in introducing modular explicit control mechanisms as in meta-rules, ATNs, and agent-systems.

¤ **Knowledge acquisition.** There are several reasons for the fact that knowledge acquisition constitutes by far the most expensive part in the development of an expert system:
(1) Human expert knowledge is primarily heuristic knowledge with hardly any explicit documentation being available.
(2) Human expert knowledge is efficiently available whenever there is a need to apply it. This implies that acquiring such knowledge is best done by observing an expert when applying his knowledge. This is done by a **knowledge engineer** interrogating the expert and transforming answers and observations on answers into representations for the XPS.

Hence, knowledge acquisition requires heavy participation of human experts and knowledge engineers. There are two techniques for **mechanizing resp. supporting knowledge acquisition:**
¤ **Inductive inference,** as done in e.g. grammatical inference +BIE 766 program synthesis from examples and forming cartesian

covers.

- **XP-dialogue in a generate- and test-cycle:** A PDIS generates examples and presents them to an XP for judgment. If the XP declines, the PDIS explains its reasoning to the XP, and XP and PDIS cooperate in correcting the PDIS. This technique is realized e.g. in TEIRESIAS [DAV 79] and APE [BOR 81].

5. Conclusions and Future Trends

5.1. Satisfying Demand

(1) Cost-profit analyses have not been made so far.
(2) XPS provide the only mechanized support for mastering diffuse fields. An XPS constitutes both a theory of a diffuse field and a tool for mechanizing the application of expertise. Given the slow development of diffuse fields, we expect XPS to inroduce a major thrust towards speeding up the rate of innovations in such fields.
(3) Measured in terms of performance, XPS belong to the most successful AI-systems.
(4) XPS contribute mechanisms towards mechanizing the following intellectual activities:
   - knowledge-acquisition.
   - abstraction.
   - knowledge-representation.

5.2. Current State

(1) Current XPS provide mechanized consultance for expert clients. They are not capable of consulting non-experts reliable.
(2) Advanced XPS exhibit competence on small , well-confined fields which exhibits that of human specialilsts.
(3) The expense for developing competent, advanced XPS is of the order of men-decades.
(4) XPS command representing and making available large amounts of small, isolated chunks of knowledge.
(5) Unsolved problems are:
   - knowledge acquisition.
   - techniques for making XPS cooperate with exparts and clients efficiently.

5.3. Future Trends

(1) XPS require vast and fast memories. XPS are less processor-intensive and more memory-intensive.
(2) Technologies for data bases and information systems become a sub-area of knowledge engineering. There is a danger that this will be recognized by research funding agencies too late.
(3) A next step in the evolution of XPS will be concurrently cooperating pattern-directed inference systems realized on multicomputers.

6. Bibliography

[BAR 79] Barstow, D.R. An experiment in knowledge-based automatic programming.A J. Artificial Intelligence: 12(1979)73-119.

[BCEM 78] Bennet,J., Creary,L., Englemore,R.S., Melosh,R. SACON: a knowledge-based consultant for structural analysis. Memo HPP-78-28/Stan-CS-78-699. Stanford Univ., Dept. of Computer Sci., 1978.

[BF 78] Buchanan,B.G., Feigenbaum,E.A. DENDRAL and META-DENDRAL: Their applications dimensions. J. Artificial Intelligence: 11(1978).

[BIE 76] Bierman,A.W. Approaches to automatic programming, in Advances in Computers (eds. Yovits, Rubinoff), Vol.15. Academic Press, 1976.

[BOR 81] Bartels,U., Olthoff,W., Raulefs,P. APE: An expert system for automatic programming from abstract specifications of data types and algorithms. Memo SEKI-BN-81-01. Univ.Bonn, Inst. f. Informatik III, 1981.

[BUN 78] Bundy,A. Will it reach the top? Prediction in the mechanics world. J. Artificial Inteligence, 1978.

[DAV 79] Davis,R. Interactive transfer of expertise. J.Artificial Intelligence:12(1979)121-157.

[ET 79] Engelmore,R., Terry,A. Structure and function of the CRYSALIS system. Proc. 6th IJCAI-79, Cambridge, 1979.

[FAG 78] Fagan,L. M. Ventilator management: a program to provide on-line consultative advice in the intensive care unit. Memo HPP-78-16. Stanford University, Dept. of Computer Sci., 1978.

[FEI 80] Feigenbaum,E.A. Expert Systems - looking back and looking ahead. Proc. 10.GI-Jahrestagung (Saarbrucken 1980), Springer Informatik-Fachberichte vol.33.

[FRI 79] Friedland,P. Knowledge-based hierarchical planning in molecular genetics. Ph. D. Thesis, Stanford Unif., Computer Sci. Dept., 1979.

[FRV 81] Fischer,H.L., Raulefs,P., Voss,H. CSSA: Design and implementation model for a programming language for asynchronous concurrent processes. SEKI-Memo. Univ. Bonn, Inst. f. Informatik III, 1981.

[HDE 78] Hart,P.E., Duda,R.O., Einavdi,M.T. A computer-based consultation system for mineral exploration. Tech.Rept., SRI International (1978).

[HEW 72] Hewitt,C. Planner: A language for ... Tech.Rept. TR-258. M.I.T.A.I.-Lab., 1972.

[HO 80] Huet,G., Oppen,D.C. Equations and rewrite rules: a survey. Tech.Rept.CSL-111, SRI International (1980).

[KUN 78] Kunz,J. A physological rule-based system for interpreting pulmonary function test results. Memo HPP-78-19. Stanford University, Dept. of Computer Sci., 1978.

[MC 78] Michalski,R.S., Chilansky,R. Knowledge-acquisition by encoding expert rules versus computer induction from examples. Int. J. for Man-Machine Studies (1979).

[MIC 79] Michie,D.E. (Editor). Expert Systems in the Microelectronic Age. Edinburgh Univ.Press 1979.

[MIC 80] Michie,D.E. Knowledge-based systems. Tech.Rept. UIUDCS-R-80-1001. Univ. of Illinois (Urbana), Dept. of

Computer Science, 1980.

[PMM 77] Pople,M.E., Myers,J.D., Miller,R.A. DIALOG: a model of diagnostic logic for internal medicine. Proc.5th IJCAI-77, Cambridge 1977.

[SHO 76] Shortliffe,E.H. Computer-Based Medical Consultations: MYCIN. Elsevier/North-Holland Publ. Co., 1976.

[SS 77] Stallman,R.M., Sussman,G.J. Forward reasoning and dependency-directed backtracking in a system for coputer-aided circuit analysis. J. Artificial Intelligence: 9(1977)135-196.

[STE 78] Stefik,M. Inferring DNA structures from segmentation data. J. Artificial Intelligence:11(1978)85-114.

[SWA 77] Swartout,W.R. A Digitalis Therapy Advisor with explantations. Tech.Rept. MIT/LCS/TR-176, M.I.T., 1977.

[WAH 80] Wahlster,W. Theorie, Entwurf und Implementation einer Erklarungskomponente fur approximative Inferenzprozesse in naturlichsprachlichen Dialogsystemen. Dissertation. Univ. Hamburg, Fachbereich Informatik, 1980.

[WAR 77] Warren, D.H.D. Pereira,C.M., Pereira,F.C.N. Prolog- the language and its implementation compared with Lisp. Proc.ACM Symp. on AI and Programming Languages, Rochester, 1977.

[WHR 78] Waterman,D.A., Hayes-Roth,F. (Editors). Pattern-Directed Inference Systems. Academic Press, 1978.

[WIP 74] Wipke,W.T. Computer-assisted 3-dimensional synthetic analysis, in Computer Representation and Manipulation of Chemical Information, eds. Wipke, Heller, Feldmann and Hyde. Wiley Interscience, 1974.

[WOO 70] Woods,W.A. Transition Network Grammars for Natural Language Analysis. CACM:13(1970)591-606.