CANADIAN THESES ON MICROFICHE

THÈSES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Canada

NL 339 (r. 86/09)

A PROLOG-BASED TOOL for DEVELOPING a HIERARCHY of EXPERT SYSTEMS in REMOTE SENSING

by

Francois BRAUN

A thesis
presented to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Applied Sciences
in
Electrical Engineering

OTTAWA, ONTARIO, CANADA, 1985

A mes parents

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please, sign below, and give address and date.

ACKNOWLEDGEMENTS

Dr M. Goldberg est specialement remercie pour son aide constante et sa patience durant la definition du projet et durant la redaction de la presente these. C'est aussi par son intermediaire que j'ai pu passer deux ans a decouvrir l'Amerique du Nord. Je tiens aussi a remercier les autres membres du departement, auusi bien les collegues de travail que le personnel administratif dont la gentillesse ne s'est pas dementie durant cette periode. Une note speciale pour le Dr Oppacher dont le cours m'a donne les rudiments de l'Intelligence Artificielle de maniere particulierement efficace. Je tiens aussi a remercier le Centre Canadien de Teledetection pour m'avoir permis de travailler avec ses moyens techniques, et en particulier, J.F. Meunier et D. Goodenough.

Un mot final est donne pour l'Entraide Universitaire Mondiale du Canada qui a fourni les moyens financiers et aussi une attention remarquable, en particulier, A.M. Bekaert et N. Clifford.

CONTENTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Expert systems are recognized to perform well in narrow domains where rules of thumb are essential to solve problems (Gevarter, 83). Expanding their domain of competence generally implies that problems of consistency of the manipulated knowledge, of control of the overall system, of combinatorial explosion of the search space are encountered (Barr & Feigenbaum, 81), (Erman & al., 80). Several organizations of expert systems have been proposed which have a number of expert systems cooperate, each managing some specialized domain, thus allowing the handling of a larger domain (Erman & al., 80), (Rychener & al., 84). However, such issues as the overall control and the coherence of the dedicated system activities are important (Durfee & al., 85).

Remote sensing, the science of acquiring data at a distance and their interpretation is a large and complex field encompassing a number of related but well-defined sub-domains (Swain & Davis, 78). An important application is the detection of changes in forested areas for resource management purposes (Park & al., 83). Automating the process of detecting changes by use of conventional and algorithmic programs have achieved only limited successes. Among the problems are the sophistication of the decision processes required, which involve heuristic decision rules rather than decision-theoretic ones, and the complexity of the context (Tinney & al., 83). It would thus be interesting to introduce expert systems in remote sensing.

The purpose of the thesis was to design a tool for the building of a hierarchy of expert systems specialized in remote sensing. SEVERE, an acronym for Systeme Expert Vide pour l'Edification d'un Reseau d'Experts (an Empty Expert System for the Construction of a

Network of Experts) is the result of the study. Moreover, to prove the feasibility of the hierarchy, SEVERE was used twice to instantiate two linked nodes dedicated to the particular domain of detecting changes in forested areas. The instantiation highlights the capabilities of SEVERE, and the promises of creating a distributed environment of hierarchically organized expert systems in remote sensing.

The present thesis will thus first recall the essential aspects of artificial intelligence and remote sensing in chapter two and three. Then a hierarchical organization of expert systems is proposed highlighting the design issues in chapter four. A presentation of SEVERE follows in chapter five describing the main design decisions made for its implementation. SEVERE is then instantiated in chapter six as an expert system dedicated to the detection of changes in forested areas. The instantiation presents the problem at hand and also the approach taken for its solution. The appendix contains examples of the output provided by SEVERE in the case of the proposed instantiation.

CHAPTER 2

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

This chapter presents a brief historical review of Artificial Intelligence, from the perspective of expert systems. The main features generally found in expert systems are introduced. Production systems are emphasized and the functioning of control structures is explained in the light of a taxonomy. We conclude with a discussion on distributed expert systems.

2.1 HISTORICAL REVIEW OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Artificial intelligence (AI) is the generic term for the field concerned with creating smart computers (Winston & Prendergast, 84). A "smart computer" is a system which could exhibit, to some degree, such "intelligent behaviors" as seeing, solving problems, understanding a natural language, learning, etc...Each of these capabilities correspond to sub-domains of Artificial Intelligence.

2.1.1 The Early Period

Artificial Intelligence emerged as a research field as soon as the first computer, ENIAC, was built in 1944. The first studies dealt with finding solutions paths in search spaces (first studies by Von Neumann and Morgenstern) (Cordier, 84). However, such studies required too high computing power for the time, and interesting studies did not begin before the late fifties. Soon, several encouraging results gave the hope that a "general problem solver" could be devised, which could solve any (or many) problem(s) by applying a few methods of reasoning and little knowledge. Research was focused on finding those methods, and did not succeed in solving other than theoretic problems. Handling real situations implied a

combinatorial explosion in the search space which ruled out actual applications. These early studies were concerned with problems with a simple wording, where everything is known with certainty, and where a deep and large arborescence had to be created (Pitrat, 84). Solving mathematical equations was a typical problem. Such problems as formulating a medical diagnosis were considered as ill-defined or ill-structured problems, with possibly erroneous data and unreliable knowledge and, thus, set aside.

In a parallel direction with this unsuccessful research, some scientists began to design systems, dedicated to a limited domain, with a simple "engine" which manipulated a large amount of knowledge stored outside the engine and provided to the system by an expert of the field. The underlying idea was that humans use a tremendous amount of knowledge in solving problems, that they are often specialized and consequently their competence degrades quickly as soon as the problem at hand goes outside the scope of their favourite domain.

The first "expert system", named "Dendral" (BUCHANAN & al, 69), was designed to find the developed structure of unknown chemical compounds from experimental data. The team which built it was initially composed of two AI-researchers and a Chemist; many other researchers afterwards contributed to increase Dendral's power.

Dendral exhibited many of the current features of expert systems:

1. The domain was very narrow.

2. Knowledge was not mixed with the mechanism making inferences, and was stored separately in a "Knowledge Base" (Figure 2-0).

3. Knowledge, especially of heuristic nature, was acquired from a human expert and translated into a coded form by one or two knowledge engineers.

Studies demonstrated that Dendral's power was not based upon its inference mechanism, which was fairly simple, but upon the completeness and the quality of its knowledge base (Smith & Carhart, 78).

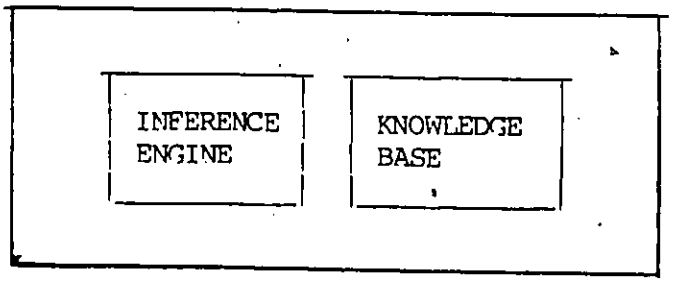

Figure 2-0
One of the Basic Concepts of an Expert System :
Knowledge and Deduction System are separated. The Knowledge Base stores the knowledge of a limited domain, which is used by the inference engine for inferring results and providing a solution to a problem.

2.1.2 The Fertile Period

Dendral showed the way for many other expert systems devised during the seventies. Further important concepts in Knowledge Engineering

emerged from experimental systems, such as, Mycin (Buchanan & Shortliffe, 84), Hearsay (Lea & Shoup, 80), Internist (Pople, 77), Prospector (Duda & al, 78), Many features found in present expert systems and the way they are implemented derive directly from these experiments. The blackboard approach, the distinction between meta- and object- levels, concepts which will be used throughout this study, are examples of the important results drawn from this period (Barr & Feigenbaum, 81), (Hayes-Roth & al., 83), (Buchanan & Shortliffe, 84).

2.1.3 Perspectives

Some evolution is forecast for the architecture and the characteristics of expert systems. The future may see the emergence of the expert systems that would include features such as (Winston & Prendergast, 84):

1. A genuine natural language interface, allowing its use by non-specialist people.

2. The capability of automatically breaking acquired knowledge into elementary pieces.

3. The capability of automatically structuring the knowledge base, thus easing the maintenance and consistency checking (Husson, 85).

4. The capability of automatically acquiring knowledge from experience, i.e. learning.

5. The capability of determining the relevance of a request: is it within the scope of the system's competence ?

6. The capability of degrading gracefully when the request is at the limit or outside of the system's competence.

2.2 RELATED CONCEPTS

2.2.1 Artificial Intelligence Programming Languages

One important design issue in Artificial Intelligence is the developing of adequate programming languages. One of the main advantages of these languages over conventional ones is their ease in manipulating symbolic data rather than only numerical data. The first such language was IPL (Newell, Shaw & Simon, 57), which is no longer used. One of the most popular languages is undoubtly Lisp (Winston, 77) which stands for List Processing, and has the distinctive capability of using lists, for both writing programs and storing data. This unique feature allows a program to reason notably on data, but also on itself. Other new features such as the recursive use of conditional expressions, a more declarative or mathematical way of representing functions than the step by step representation of most languages, also contributed to the success of Lisp, especially in the United States. However, a few other languages are also popular, such as Pop-2 (Burstall & al., 71) mostly used in Great Britain and Prolog in Europe and Japan. Prolog, which stands for "Programming in Logic", was proposed by A. Caulmerauer in 1969 (Roussel, 75), and is a sub-set of the different concepts used in first-order predicate-logic (Grumbach, 82). Prolog allows recursions, and backtracking in a

search space, and has an embedded inference engine, "the Prolog proof procedure" (Clocksin & Mellish,81). Such features make Prolog a language of choice for the design of expert systems (Colmerauer, 84), (Pereira & Warren, 80) and a number of expert systems were recently written in Prolog (Parcy, 82) or in languages related to the Prolog family such as SNARK (Lauriere, 84), (Popescu, 84). M-Prolog is a dialect with interesting features such as many built-in predicates easing the writing, the capability of creating modules and of completely defining the links between the modules (Gur-Arie, 84).

Other languages have been designed either as expert-systems building systems such as Ops-5 (Forgy & McDermott, 77) or as powerful languages for specialized domains such as Smalltalk in education (Kay & Goldberg, 77). Last but not least, most AI-oriented languages are proposed with powerful support environment easing the writing, debugging and testing of implementations.

2.2.2 Expert Systems

An expert system is a program intended to mimmic a human expert's way of reasoning within a specialized and delimited domain (Bramer, 82). It is generally understood that expert systems have the following capabilities (Buchanan, 82):

a) Expert systems model human knowledge in a limited domain and apply it to solve problems.

b) Expert systems handle uncertainty by storing uncertain knowledge and by being able to combine uncertain hypotheses.

c) Expert systems provide explanations about the line of reasoning.

d) Expert systems can modify the knowledge-base by learning or by direct acquisition of new pieces of knowledge from a human expert.

Expert systems contain two types of knowledge (Cordier, 84):

1) General knowledge of the domain, the one which can be found in relevant books, and is usually of a factual nature.

2) Heuristic Knowledge, that is knowledge which can only be acquired from experience. This knowledge is also called "rules of thumb" and may be subject to dispute.

It is the ability of expert systems for storing, exploiting and explaining heuristic knowledge which makes them valuable.

2.2.3 Knowledge Engineering

Knowledge Engineering is the field where knowledge-based sytems, i.e. expert systems, are studied and applied. A knowledge engineer creates the "inference engine", i.e. the system which will manipulate knowledge to make deductions, and also translates in a form understandable by an expert system, knowledge provided by a human expert. Moreover, the knowledge engineer also generally designs the other systems required in an expert system, namely the explanation facility, the knowledge acquisition system, and the natural-language

interface.

2.2.4 Knowledge Representation

The history of Artificial Intelligence shows that knowledge is the key issue in the design of expert systems and the way it is stored is of prime importance. Such questions as "how to precisely enter an expert's idea", "how to ease the manipulation of the knowledge by the expert system", "how knowledge can be easily altered or updated", "how to enter such features as time and uncertainty",... are considered in the choice of the representation.

2.2.5 Rule-Based Expert Systems

Many ways of encoding knowledge have been proposed (see '2.4'), but most existing expert systems use a rule format for this purpose. A rule is a "situation-action" pair, the situation being a description, comprising both elements that may exist and conditions, the action being a set of deductions. Rules were already used by Babylonians to "rule" everyday affairs (Jaynes, 76). The following is an example of one rule:

Situation : if a horse enters a man's house
and (if) it bites either an ass or a man,

Action : (then) the owner of the house will die
and his household will be scattered.

A set of rules is also called a "production system" (production meaning rule here). The inference engine of a rule-based expert system thus combines the rules which may be fired to reach conclusions

or solutions of a given-problem.

2.3 EXPERT SYSTEMS: A DETAILED PRESENTATION

2.3.1 General Architecture

An Expert System generally contains the following five elements (Figure 2.1) :

1. A Knowledge-base which stores all the different kinds of knowledge the system embodies.

2. An Inference Engine which, from the available knowledge and from entered data about the problem at hand, makes deductions and provides an answer to the problem.

3. A user interface which allows the user and the expert system to communicate.

4. An explanation facility which provides explanations about how the system reasons on a given problem. The explanation facility interacts with the inference engine and the knowledge-base to answer requests. Communication with the user is usually performed through the user-interface.

5. A knowledge acquisition system which allows a human expert, through the user-interface, to access and change the knowledge base.

The first two modules compose what is called "the shell".

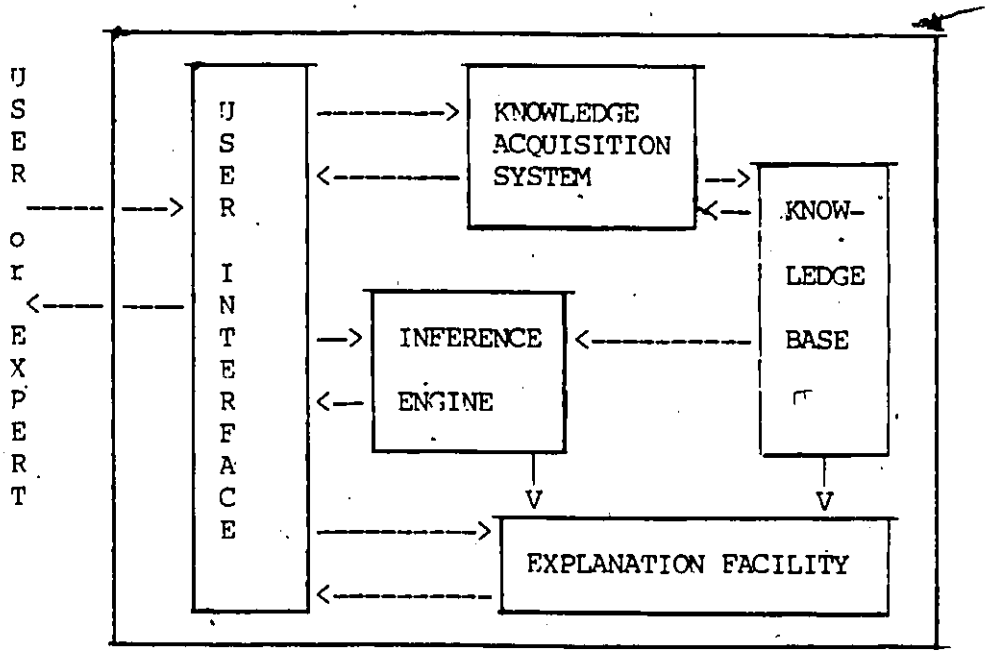

Figure 2-1
General Structure
of a stand-alone Expert System including
a shell composed of the inference engine
and the knowledge base. The arrows
represent the sending of commands and/or data

Though not explicitly shown in Figure 2-1, a knowledge engineer is required to interface the domain expert with the system. It is generally recognized that the present capabilities of natural language interfaces are not powerful enough for a general user to interact with a knowledge acquisition system or an explanation facility (Grumbach, 82).

2.3.2 Architecture Of The Shell

The discussion below is restricted to expert systems dedicated to "classification problems" or diagnosis. In these problems, the exhaustive list of all the possible solutions are known beforehand and

the purpose of the expert system, therefore, is to select the best solutions from the list.

The proto-typical shell architecture shown in Figure 2-2, is derived from a schema in (Hayes-Roth & al., 83) and is based upon a blackboard for communications. The main elements are as follows :

1. A Knowledge-base containing knowledge about the domain of interest and also about the way of solving problems.

2. An Interpreter which applies the knowledge to the problem at hand, and updates the confidence the system has about the results it found so far.

3. A Scheduler which elaborates and executes a plan of actions for reaching a solution. The plan typically contains an ordering of elements of the system to call. The scheduler decides, for instance, when to call the interpreter, and defines to the interpreter what it has to execute.

4. An Arbitrator which resolves conflicts. A conflict arises whenever contradictory results are found. For instance, when a problem may have several competing and contradictory solutions, the arbitrator is typically called to select the likeliest solution among those deduced during the session.

5. A Justifier Interface which stores all the needed information for the explanation facility. This information will be used by the explanation facility to retrieve the line of reasoning used by the shell and explain each step and deduction made during the session.

6. A Blackboard which contains

- the data known about the problem at hand,
- the (intermediate-)results found so far (in subdivision "Data & Results Store"),
- the final result or solution (in subdivision "Solution"),
- the steps of the plan elaborated by the scheduler (in subdivision "Agenda").

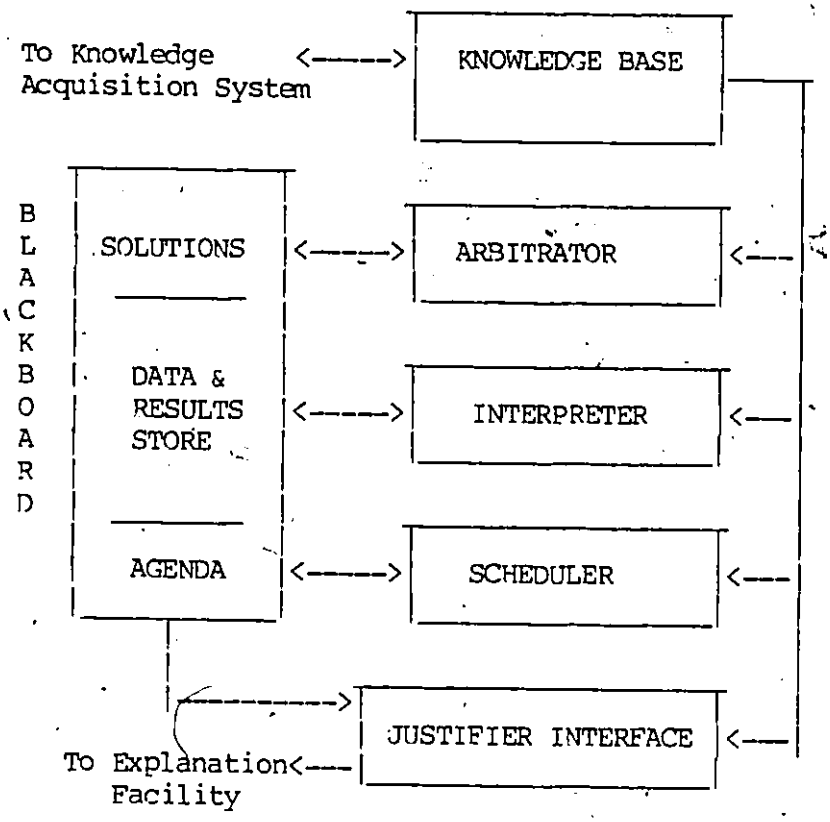

Figure 2-2
Anatomy of a Typical Shell
with a Blackboard and a possibly existing
interface with outside (expert) systems

The scheduler may contain two levels, in which case one is dedicated to the design of a plan of action and is called the meta-level, and the second to its execution. If this two-level architecture is chosen, the second level is generally called the scheduler (though an inconsistency arises) and the first-level is then seen as an independant module (called the "meta-level" or "meta-expert"). Some expert systems contain a third level called the "meta-meta-level" which contains knowledge about the manner for designing plans (Davis & Lenat, 82). The design of a meta-level is recommended whenever an exhaustive search becomes unfeasible and is used to guide the search through the more promising lines of reasoning. This is accomplished by focusing on the more interesting paths in the search space and pruning those paths which are less promising. An example of a meta-level is given in chapter six.

Not all existing expert systems use a blackboard to store plans, data or results. However, the blackboard is usually considered as a more general way of storage than other techniques, such as the context tree (Buchanan & Shortliffe, 84), or the semantic net (Dudat & al., 78). In the latter, the manner for representating knowledge is fixed, whereas no such assumption is made or needed inside a blackboard.

2.3.3 Architecture Of The Other Modules

The knowledge acquisition system and the explanation facility handle tasks which are at present only superficially understood: "Our efforts at explanation and knowledge acquisition have only scratched the surface" (Winston & Prendergast, 84). Natural language processing is beyond the scope of the present study and will not be actually

considered.

(Davis & Lenat, 82) proposes a schema of what could be a general architecture of an explanation facility which is highlighted in Figure 2-3:

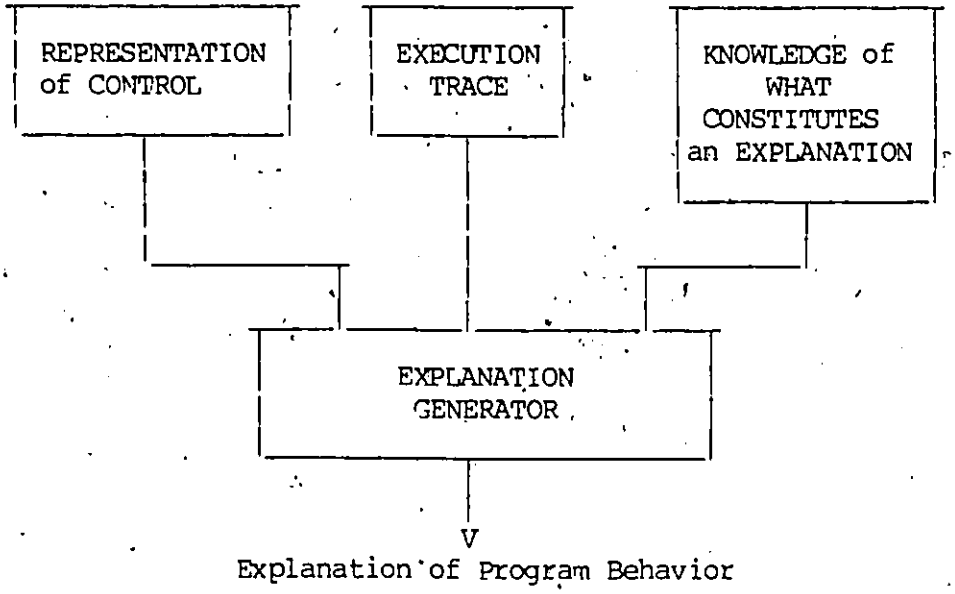

Figure 2-3
Example of a General Architecture of an Explanation Facility, according to (Davis & Lenat, 82)

The Execution Trace is a list of records about a session of an expert system, which stores the necessary information to highlight the steps in the reasoning, and how and which elements were used or deduced.

The Knowledge of What Constitutes an Explanation controls what should be put into an explanation, and how it should be presented.

The Representation of the Control Structure models the way the expert system makes deductions, how knowledge is manipulated and in which order. The representation of the control structure can be, therefore,

considered as a summary of the characteristics of the inference engine given to the explanation facility.

Expert systems capabilities for providing explanations are very restricted. Such parameters as, the user's level of understanding of the knowledge base; of the expert system itself; the user's level of familiarity about the studied field; or his/her expectations about explanations; are arbitrarily set during the design and generally do not change. In other words, the facility does not adapt itself to the user's capacity (Oppacher, 84).

However, a number of implementations show interesting abilities, and, in particular, Teiresias (Davis & Lenat, 82). Teiresias is a superset of Mycin (Shortliffe, 76) which is an expert system dedicated to the diagnosis of infectious blood diseases. Teiresias contains, apart from the shell (i.e., Mycin), a "user interface", an "explanation facility", and a "knowledge acquisition system" which handle well their respective tasks and its main ideas of implementation are used in this study.

2.3.3.1 The Explanation Facility Of Teiresias

Teiresias basically answers why-type and how-type questions which can be stated as follows :

1. why the system decided to use a particular rule (or meta-rule).

2. why the system decided to use a particular element.

3. how the system used a particular rule (or meta-rule).

4. how the system used a particular element.

A request about an element is handled according to the type of the question. Why-type questions explain the situation in which the system used an element to infer another element. The facility highlights how the inference engine went one step farther in its deductions, by use of the element. How-type questions have an opposite meaning in the sense that, given an element, they ask how the inference engine arrived at that element. Here, the system shows the previous situation, from which the inference engine made a one-step deduction to obtain the given element.

For explanations about rules, the system merely presents the targeted rule, its instantiated premise and conclusion, and the values of the certainty factors.

2.3.3.2 The Knowledge Acquisition System Of Teiresias

Teiresias uses models of rules to guide the acquisition of new rules. A model is a set of characteristics which are common for a set of rules. By comparing a newly entered rule with the relevant model, the system is able to warn or advise the user whenever large discrepancies are found between the rule and what "it should be" according to the model. Rules having the same type of action-part are represented by one model. Models are organized in tree structures. The root-model is the more general one. Son-nodes are refinements of

the father-node. For instance, a model about rules concluding about "changes" in remote sensing, may have two sons, one model for the rules confirming a change, the other for those invalidating a change. The third level nodes may contain models representing rules which confirm (or invalidate, according to the father-node) a certain type of change.

A model in Teiresias contains the following sets:

1. the set of the rules represented by the model,
2. the set of conclusions that may be found,
3. the set of elements in the premises,
4. the set of correlations existing between two different conclusions,
5. the set of correlations existing between two different premise elements,
6. one pointer to the father-node, if it exists,
7. one pointer to the son-nodes, whenever such nodes exist.

Each listed conclusion, element of a premise, and correlation is assigned a number corresponding to a frequency of occurence within the represented set of rules (the actual number being a weighted frequency).

Figure 2-4 is an example of a model for the following two rules:

1. Change <--- A and B and C

2. Change <--- A and B and D

Sets of the Model	Content of the Sets	Frequency
set of rules	1,2	-
set of conclusions	Change	100%
set of premise elements	A	100%
	B	100%
	C	50%
	D	50%
set of correlated - actions	-	-
- premise-elements	A and B	100%
	A and C	50%
	A and D	50%
	B and C	50%
	B and D	50%
pointer to father-node	-	-
pointer to son-nodes	-	-

Figure 2-4
A model of rules used in
Teiresias

To further illustrate the concept, suppose that the following new rule is proposed to Teiresias:

Change <--- A and E

The first step is to find the appropriate model. The system recursively tries to find a model fitting the rule, by going deeper and deeper into the model-tree. Each time a model fits and includes a pointer to more specialized models the new models are studied in turn.

The last model which fits is then selected. The comparison rule/model is performed by looking for an inclusion of the set of actions of the rule with the set of typical conclusions of the model. In the example, only one model exists and fits the rule as it has the same action part, "Change".

Teiresias now compares the premise with the set of typical premise elements. The elements of the set which are not contained in the rule are presented to the user, as long as their frequency of occurrence is above a predefined threshold. The user may then decide to keep or reject these elements. For example, a threshold of 75% is set, and B will then be proposed to the user. The same step is now performed on the action-part of the rule. In the example, since no other conclusions are made by the rules of the model, no other elements are proposed.

Teiresias now attempts to find missing correlations in the premise part. If an element is present in the rule and if there exists a correlation between the element and another which is not in the rule, then the user is informed. The system can be asked to guess the missing elements, by use of the correlation in the model and the context. For example, if the threshold for the correlations were reset to 40%, then all five correlations are presented to the user. We suppose the user decided that only the correlation A and C has to be retained, so the rule is now:

Change <--- A and E and C

The same step is now performed on the action-part of the rule but, in the example, no correlation is recorded.

The rule is then stored and the models are automatically updated. The rule number is 3, and Figure 2-5 shows the transformed model:

Sets of the Model	Content of the Sets	Frequency
set of rules	1,2,3	-
set of conclusions	Change	100%
set of premise elements	A	100%
	B	66%
	C	66%
	D	33%
	E	33%
set of correlated		
- actions	-	-
- premise-elements	A and B	66%
	A and C	66%
	A and D	33%
	A and E	33%
	B and C	33%
	B and D	33%
	C and E	33%
pointer to father-node	-	-
pointer to son-nodes	-	-

Figure 2-5
The updated model of rules
from Figure 2-4, once a new rule
is entered

2.4 PRODUCTION SYSTEMS AS KNOWLEDGE REPRESENTATION

(Barr & Feigenbaum, 81) lists seven different types of knowledge representation, the most prevalent being the production system. This study is limited to this type of representation. The following four features characterize production systems:

1. They are modular since a rule can be added, altered, or deleted without directly influencing others.

2. Human experts tend to describe their knowledge in terms of rules, thus rules are a natural way of expressing knowledge.

3. If knowledge about a domain is embodied by rules, then the control knowledge may be hard to encode. Furthermore, the control flow during a session may be hard to follow, harder than a conventional program where the sequence of steps is predetermined.

4. The absence of structure within a knowledge base allows inconsistencies to arise especially as the knowledge base becomes large. For example, the ordering of the rules may have a baneful influence and contradictory rules may exist at the same time.

The first two points explain why production systems are used so often. They are considered as a compromise between the ease of representing knowledge and the richness of the semantic links which exist between elementary pieces of knowledge.

2.5 THE DIFFERENT TYPES OF CONTROL STRUCTURE

To highlight the main characteristics of control structures and to show the available options in the design of inference engines, we use the classification and terminology proposed by (Laurent, 84). The control structure of an expert system deals with the way inference engines function, i.e., in the case of rule-based systems, how they fire rules.

2.5.1 Definitions

The terminology used in this part includes:

1. An object : an element of the knowledge base which represents a factual interpretation of the corresponding abstract knowledge.

2. An action : an element of the knowledge base which represents a deductive interpretation of the corresponding abstract knowledge. A rule is an example of an action.

3. A state : a set of data and knowledge available for the problem at hand. The "current state" is the current set of data and knowledge. At the beginning, the initial state is also the current state. A state is also considered as a particular situation of the work space. The development of the work space is then seen as the expansion of a tree whose nodes represent different states. The current state is represented by the last node reached. If the inference engine always starts from the last node reached to expand the

work space, it is said that the current state never changes. If the inference engine always selects a starting state prior to expansion, it is said that the current state can change. For instance, the inference engine may decide to restart from the initial state if a dead-end is reached.

4. A transition : the use by the inference engine of an action which modifies the starting state into a new one, thus expanding the work space. A transition can be represented by a couple (Object, Action), with the Action being performed on the Object.

To highlight these concepts, we suppose the knowledge base is composed of:

A <--- B and C :R1

A <--- B and E :R2 D, E: facts

B <--- D and E :R3

We suppose further that D and E are known, that A is to be proved, and that the inference engine functions in backward chaining, (a depth-first search starting from the goal is executed). We finally suppose that the inference engine never changes its current state. The work space is thus expanded as presented in Figure 2-6:

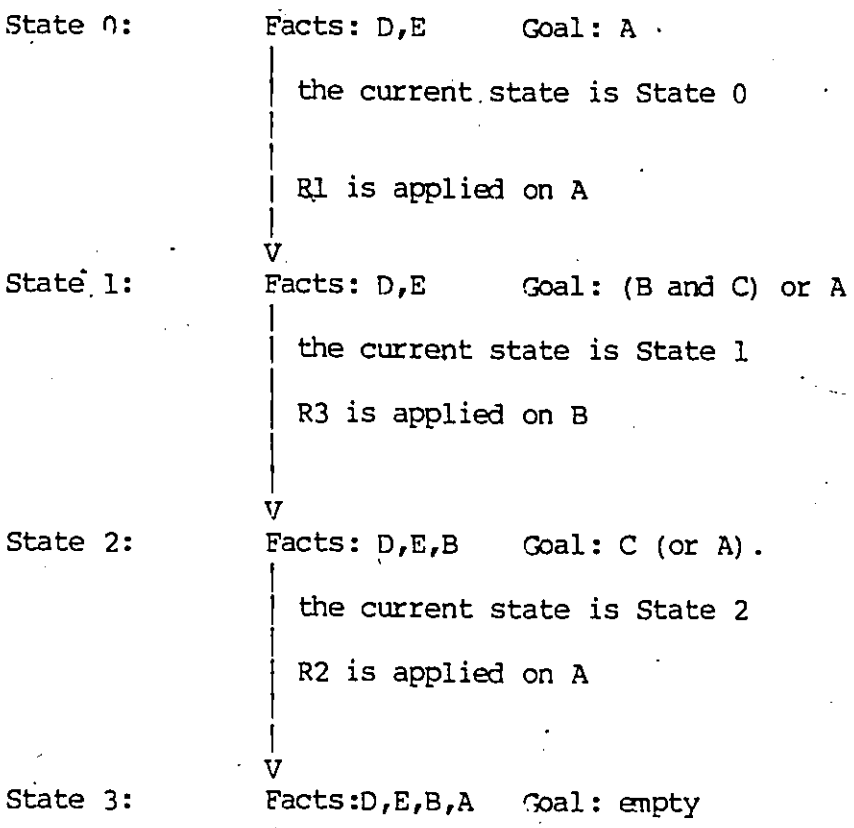

Figure 2-6
Example of a Inference Engine
which never changes the current state
since it is always represented by the
last reached state

In state 2, the system fails to prove C so the option is to prove A.

We now suppose that the inference engine has the capability of changing the current state, and that it knows that C cannot be proved. By noticing C at state 1, the inference engine will take the (sole) option of reinstalling state 0 in the work space. Figure 2-7 shows the different states induced by such an inference engine:

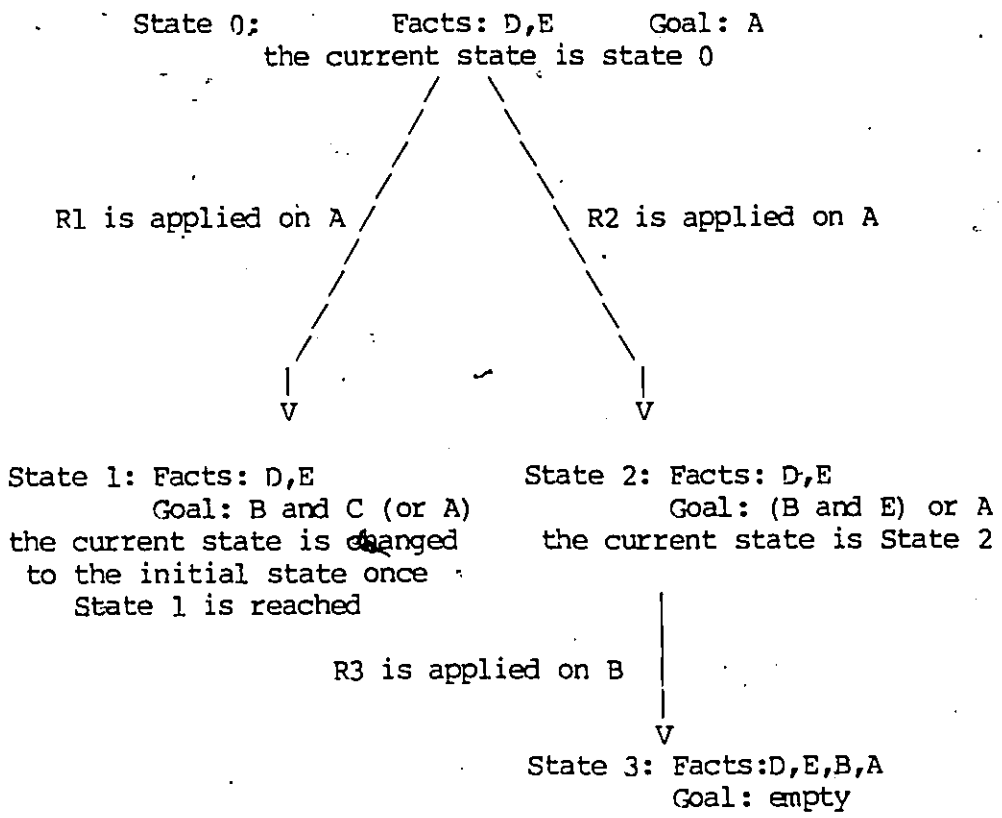

Figure 2-7
Example of a Inference Engine
which can change the current state. State 1
being recognized as not promising, State 0
is reinstalled in the work space, leading
to state 2 and then state 3.

2.5.2 Elementary Taxonomy Of Control Structures

Classification problems are often seen as a search in a work space which is represented by the tree of all the possible states. The root node is the initial state, characterized by the initial data and the request. A leaf is a final state, or an "answer" to the problem at hand. The system passes from one state to another until a final state is found. The passage from a state/node to another can be done in several ways. The following list explicits the alternatives by three

main design options corresponding to the three levels of Figure 2-8:

1. If a prior selection of the starting node is performed from among all the already created nodes, there is a change of "current state" (type S). If the starting node is always the last node reached, then there is no change of "current state". Allowing the current state to be changed or not is the first design option.

2. Apart from choosing the starting node, the inference engine has several ways of selecting the transition, i.e. the couple composed of an object to process and an action to apply on the selected object. The couple may be selected simultaneously as in Ops (Forgy, 79) or Snark (Lauriere, 82). The type of these engines is C, where C stands for "Conflict". The term "Conflict" is used by reference to the classical definition of a conflict in expert system: a conflict is said to arise in an inference engine when several couples may be chosen at the same time as a transition (Laurent, 84) (Forgy, 78). This happens, for example, when several rules are firable simultaneously. Another choice is to select the couple in two steps. The order may be to choose the object to process and then an action applied on the object (type O-A) (O for Object, and A for Action). The reverse, choosing first an action and then the object to process by the action, is performed by an inference engine of type (A-O).

Six different types of engines can thus be identified according the taxonomy, which are the leaf-nodes of the following tree:

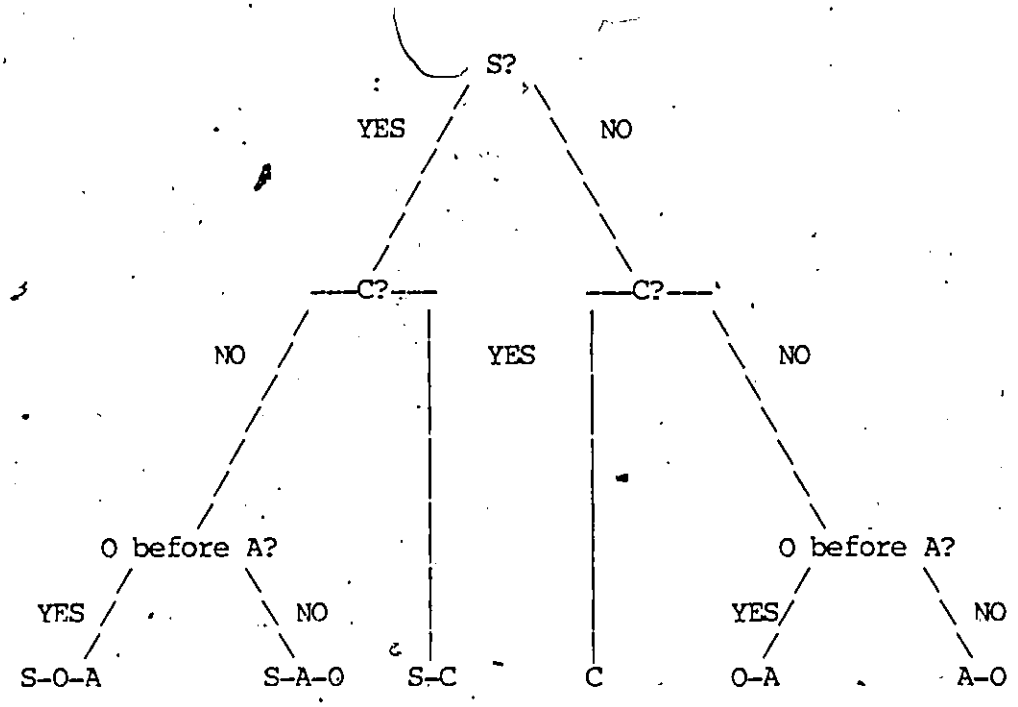

Figure 2-8 : six types of control structure

S stands for :change of current state
A stands for :prior selection of action
O stands for :prior selection of object
C stands for :simultaneous selection of action and object

Existing expert systems can be classified into one of the given categories. For example, the language Prolog (considered as an expert system), is of the type S-O-A, since the current state may be changed by backtracking, and since it selects the first encountered predicate for processing. Emycin, on the other hand, is of type O-A, since no change of current state is performed, and since an object is first systematically selected as the last generated goal-object followed by the choice of an action.

2.5.3 Backward Or Forward Chaining

In (Winston, 77), two types of control structure are proposed:

1. The backward chaining approach also called "goal-driven approach".

2. The forward chaining approach also called "data-driven approach".

A third approach which is a mixture of the precedents is described in (Barr & Feigenbaum, 81).

In forward chaining, the inference engine first considers the available data and by successively firing rules and inferring results, reaches the goal. The knowledge of the goal only helps in deciding when to stop.

Backward chaining is obtained by executing the reverse process: the goal is considered first. A rule drawing conclusions about this goal is found and fired by the inference engine. The goal is then replaced by the premise elements of the chosen rule; these elements now becoming the current goal. The engine iterates the process until the current set is directly proved by comparing it with the available data, or until no more rules can be used. In the latter case, the initial goal is found not to be true, either because the goal is false or because the data are not sufficient. Backward chaining is a depth-first search applied on a production system.

Choosing between Backward or Forward chaining is a matter of selecting the object to process respectively from the current goal set or the data. The two modes of chaining are now seen as particular features of an engine and will not change its type.

2.6 HANDLING OF UNCERTAINTY

2.6.1 Importance Of Handling Uncertainty

Since the value of an expert system is directly dependent upon the heuristics it contains and since, by definition, heuristics are not always true it is essential to provide some measure of how often a rule is true, and to provide some way of combining intermediate, uncertain results. Uncertainty is related to the degree of hesitation an expert has for a given problem, and to the degree of confidence the expert has in the solution he/she provides. A measure of confidence should thus give an indication of "how promising" an hypothesis is; should allow the dynamic ordering of the hypotheses; and thus permit the system to focus on the most promising hypothesis.

The knowledge engineer is more interested in a measure which consistently ranks the hypotheses in the right order, by comparing the associated measures of confidence, than in getting an absolute value of the "probability" an element is true. Such an absolute value would be elusive since heuristic rules represent a qualitative line of reasoning rather than a rigourous approach. Moreover, any measure provided by experts would be dependent upon the experts themselves.

2.6.2 Mycin And Others Ways Of Handling Uncertainty

, Several theories handling uncertainty, have been studied in the context of Artificial Intelligence. Among them, Bayesian decision-theory (Grojnowski, 81), Fuzzy-set theory (Zadeh, 79), and Evidence theory (Shafer, 76), have a theoretic foundation. Bayesian decision-theory has been shown to be unwieldy in general (Buchanan & Shortliffe, 84), whereas the other two have yet to be confirmed through extensive experiments. Another framework, the Mycin's theory of Belief, has been designed experimentally, but later research has shown relations between Mycin and the Evidence Theory (Buchanan & Shortliffe, 84). Despite its lack of theoretic foundation, the validity of Mycin's theory is supported by the consistent results obtained in several implementations for different application domains (Kunz & al., 78), (Mulsant & Servan-Schreiber, 84), (Bennett & al., 78).

2.6.3 Mycin's Theory Of Belief

Each of the elements manipulated by rules are assigned a measure of belief and a measure of disbelief. Each measure is a number between 0 and 1. Intuitively, the measure of belief represents the total amount of confidence that the element is "true" or valid. The measure of disbelief represents the total amount of doubt the system has about the element. Both measures are independantly computed. The difference, called the "certainty factor", belonging to the interval [-1,1], represents the degree of confidence the system has that the element is "true" (if positive), or "false" (if negative).

2.7 DISTRIBUTED EXPERT SYSTEMS

Figure 2-1 shows the schema of a stand-alone expert system. Recently, several studies highlighted the potentials of cooperating expert systems (Erman & al., 80), (Nagao & Matsuyama, 82), (Rychener & al., 84), (Durfee & al., 85). When the domain of interest is large enough to encompass several interacting sub-domains, specialized expert systems communicating with one another are warranted. The sub-domains may overlap, leading to expert systems which are redundant to a certain extent. In a distributed environment, expert systems communicate through interfaces and coherently cooperate to the common design of a final solution. Key issues that need to be addressed in using distributed expert systems include: the control of the overall system, the attribution of roles for each expert system, as well as the degree of sophistication and of freedom allowed to each expert (Durfee & al., 85).

2.8 SUMMARY

The objective of the present chapter was to present the elements constituting an expert system. In the present study, the emphasis is on the shell rather than the other modules, namely the explanation facility, the knowledge acquisition system and the user-interface. The first two facilitie are nonetheless examplified by Teiresias which will be the reference design in chapter five. The shell was further presented as well as a taxonomy of inference engines. The main issues in case of communicating experts were then noted, and will be addressed in chapter four. It was pointed out that distributed expert systems are needed for large domains including a number of fairly

self-contained sub-domains. The following chapter will present some aspects of remote sensing and will highlight the potential role of knowledge-based systems for the automatic handling of tasks in remote sensing.

CHAPTER 3

CHANGE DETECTION TECHNIQUES

FOR REMOTE SENSING

The purpose of this chapter is to focus on one of the numerous domains of applications of remote sensing: the detection of changes in forested areas by use of satellite data. We point out the existing limitations for the currently used techniques and explain why knowledge-based systems may help to design more efficient systems in remote sensing.

3.1 INTRODUCTION TO REMOTE SENSING

Remote Sensing observes objects at a distance and collects data for deducing information about them (Swain & Davis, 78). Remote sensing has long been an image-oriented field using pictures and visual analysis methods (Lintz & Simonett, 76), this mode being exemplified by aerial surveying activities using cameras and films. In recent years, digitally based methods for processing and interpreting remotely sensed data have been proposed (Swain & Davis, 78), especially with the emergence of satellite based sensors producing large volumes of data. An important application of remote sensing techniques is resources assessment for updating forest species or agricultural fields maps for monitoring and controlling the environment, for planning and forecast purposes.

In the case of satellite-based remote sensing, the main tasks to be performed are represented in Figure 3-1, (from (Lintz & Simonett, 76)), by each symbolic box:

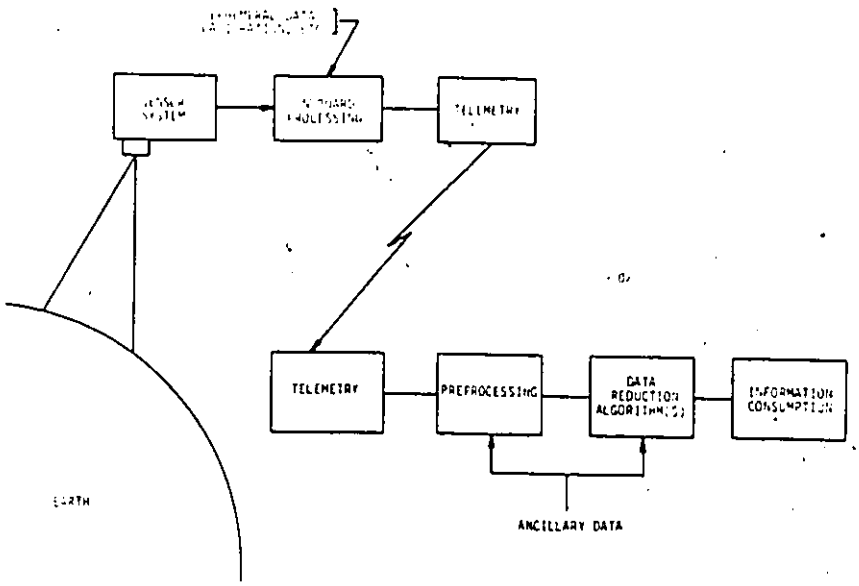

Figure 3-1

Organization of an earth survey system

Data are gathered by a sensor system, which can be a camera, and are processed for radiometric and geometric manipulation. Radiometric manipulation includes alteration of the recorded intensities for reasons of calibration, whereas geometric manipulation may be performed to correct such effects as the rotation of the earth, the parallax due to the relative positions of the satellite and the sensed ground area. Once the data are transmitted, some preprocessing and data reduction are performed. Preprocessing can include any type of image enhancement by use of transformations, spatial frequency operations, and convolutional filtering. The purpose of preprocessing

can be to enhance the boundaries of the objects included in an image or to correct transmission errors. Registration can also be performed during the preprocessing period. A newly received picture is then compared to already received pictures of the same area and transformed for an optimum matching or registration. It may also be highly interesting to reduce the amount of data to be stored by compression techniques. Finally, the received data are ready for use, i.e. for further processing or interpretation. The processing of data largely depends upon an extensive use of quantitative techniques of image analysis (Swain & Davis, 78) (Gonzalez & Wintz, 77).

3.2 SOME CHARACTERISTICS OF THE AVAILABLE DATA

The bulk of data still generally comes from aerial photography or satellite imagery taken across the available electromagnetic spectrum. Satellites are priviledged sensors, because they provide vast amounts of information on a regular basis, at a relatively low cost (Landgrebe, 81). Landsat-MSS images, which are composed of four channels, each corresponding to a spectral band where intensities record the same ground area every eighteen days at most, are a good example. However, data obtained from aerial or space-based sensors are noisy. The characteristics of the radiations, of the atmosphere, and of the sensors, imply errors in the interpretation of the data. The same object may change aspect under different lighting conditions caused by the period of the year, the day, and the weather. Different objects, such as corn and soy-bean fields, may be hard to discriminate from one another because of the similarity in reflectance value throughout the electromagnetic spectrum. A sensor always receives altered radiations of the targeted object: the sensor definition may

be insufficient to distinguish small neighbouring objects; the received radiation may be the result of the influence of the atmosphere, as a medium, of space, as a radiation emitter, of other radiation sources, espacially those of human origin; finally, the sensor itself may be faulty (Hord, 82).

Data from ground inspection are also considered but mostly for confirmation, because of their cost and irregular availability. In conjunction with imagery, many other non-pictorial data may be available, such as the sun angle, the degree of humidity, the position of the satellite or plane,the date,etc. Atthough these data may also be erroneous, they still may be employed to reduce the amount of noise in aerial and satellite pictures (Landgrebe, 81).

3.3 CHANGE DETECTION IN FORESTED AREAS BY USE OF SATELLITE DATA

Since one of the main goals of remote sensing is the timely updating of maps, change detection techniques comparing old pictures or maps with a new picture are important. Such techniques may be performed manually or automatically. However, if manual techniques generally provide a good accuracy, their systematic use is unrealizable because of the amount of data to handle (Hegyi & Quenet, 82). Extensive reviews compared many of the classical automatic techniques employed in the case of Landsat imagery (Burns & Joyce, 81), (Park & al., 82), (Braun, 84); some are presented here.

3.3.1 Mathematical Formulation

An picture, typically, one channel of a Landsat picture, is defined as an array of values of a function f having four variables:

1. i, belonging to [1,I], where I is the number of lines in the picture.

2. j, belonging to [1,J], where J is the number of columns in the picture.

3. c, belonging to [1,C], where C is the number of channels compounding the picture.

   In case of the Landsat imagery, 1 <= c <= 4.

4. t, which represents the chronogical rank the picture was taken.

In the following, comparisons are made between two pictures. We shall refer to them by T1=1 for the first picture (taken before the change period), and by T2=2 for the subsequent picture, taken after the change period.

An elementary picture, recorded at date t, will be noted { f(i,j,c,t) }, the notation meaning that the parameters i,j are within their full range interval, i.e. the number of lines and columns contained in a picture, and that c and t have a given value. A composed picture, recorded at time t,and containing the C channels will be noted { f(i,j,t) }, where i and j are within their respective full range interval, and t is given. The corresponding classified picture of { f(i,j,t) } will be noted { C(i,j,t) }.

We now present some typical change detection techniques for forested applications. We shall assume that preprocessing has been performed already so pictures are corrected, registered, normalized, etc. The result of a change detection process is typically the

labeling of the pixel locations of a picture, and a label representing a certain type of change.

3.3.2 Techniques Based On Temporal Analysis

As an example, we present the Post classification change detection technique (Swain,76), the decision process of which can be stated as follows:

if { C(i,j,1) } <> { C(i,j,2) }

then

decision = D1(C(i,j,1),C(i,j,2))

else decision = D2(C(i,j,1),C(i,j,2))

where D1 is typically a two dimensional look-up table, and where D2 generally represents the previous decision (if any) for example, or another look-up table.

The output is a decision picture created pixel by pixel.

The classification is independant of the scheme, and any classifier can be used. One of the drawbacks of the technique is the limitation in the accuracy of the results by the accuracy of the classification process itself. For example, if 2 classifications have an accuracy of 0.8, supposing the errors are randomly distributed over the 2 images, the accuracy of the change picture will be (Malila,80):

$$0.8 \times 0.8 = 0.64$$

Such techniques are highly dependent upon parameters, such as the lighting conditions, the sun angle, the nature of the ground. They may be accurate but they are characterized with a high variance in the accuracy. Limiting the variance requires some kind of adaptation, by a careful preliminary correction (Logan & Strahler, 82). Efficient and extensive corrections are not easily performed, which often prevent the operational use of trajectory-based techniques (Wheeler & Misra, 80), (Engvall & al., 77).

3.3.3 Techniques Based On Spectral Analysis

These techniques may use both:

1. spectral signatures of classes (Figure 3-2) i.e., the characteristic appearance of classes in spectral bands (Swain & Davis, 78), (Park & al., 83). The signatures can be hypothetical, highlighting what the signature of an object should be. For instance, such classes as water, clouds can be detected from satellite data by creating the practically obtained signature and matching it with the hypothetical one. Though often used in classification, spectral signatures may be used for detecting changes when a change has a typical appearance, such as the occurrence of a fire.

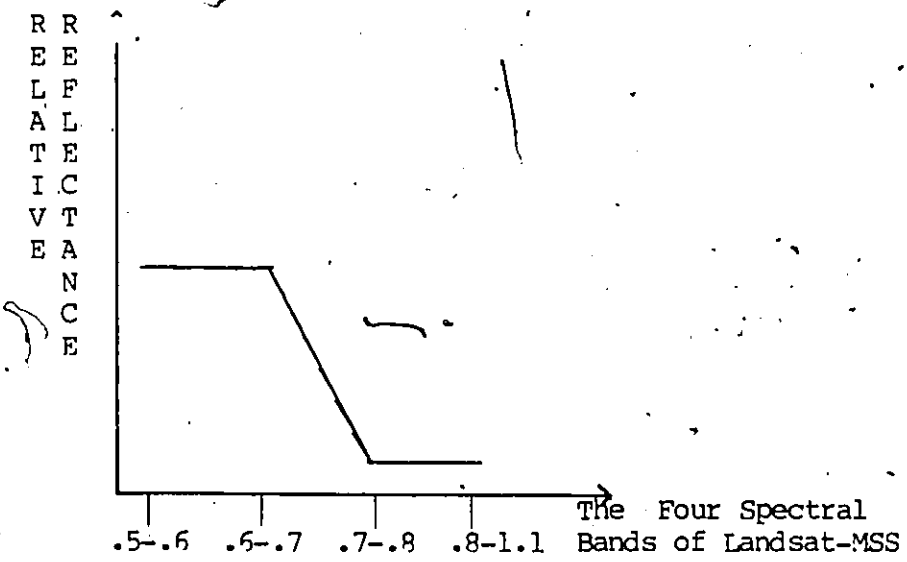

Figure 3-2
Example of a hypothetical spectral signature. The curve represents the relative reflectance of water, which is known to have a low reflectance in near infra-red

2. a statistical criterion. (Burns & Joyce, 81), (Colwell & Weber, 81), (Likens & al., 82), (Malila, 80), (Nelson, 83) present different techniques using a statistical criterion. A criterion may be the evaluation of the difference existing between the two pictures, at a given pixel location, by use of a distance. Here a threshold is used for comparison. An example is proposed by (Todd, 77), based on the use of ratios of intensity, in several spectral bands (Figure 3-3):

Two pixel intensities f(i,j,c,1) and f(i,j,c,2) are ratioed. If the absolute value of the difference: f(i,j,c,1) / f(i,j,c,2) - 1 is above a predefined threshold, a change is detected.

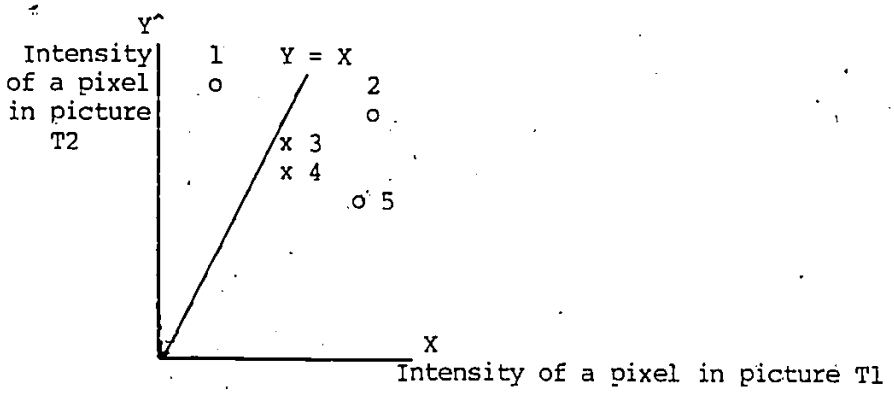

Figure 3-3
Use of a ratio-based technique to detect the pixels which changed. Two pixels at the same location have their intensity in one band ratioed and this value is thresholded. The ratio differs substantially from 1.0 for pixels 1, 2, and 5, and are labeled as changed, whereas pixels 3 and 4 are labeled "no-change".

One of the problems encountered here is the crudeness in using a threshold. Depending upon the lighting-conditions, the region, the period of the year, and so on, the threshold has to be adapted. Even in the case of an optimal threshold, mistakes are made, mainly due to the overlapping phenomenom of classes within a picture. Figure 3.4 presents two classes of pixels; they could be two types of change, which can typically be obtained with a Landsat picture (Swain & Davis, 78). The two curves are monodimensional probability density functions. The two classes form two partly overlapping clusters. By using the classical maximum-likelihood decision rule, a significant amount of pixels will be misclassified, represented in the figure by the shaded area. The amount depends upon the degree of overlap between the two clusters.

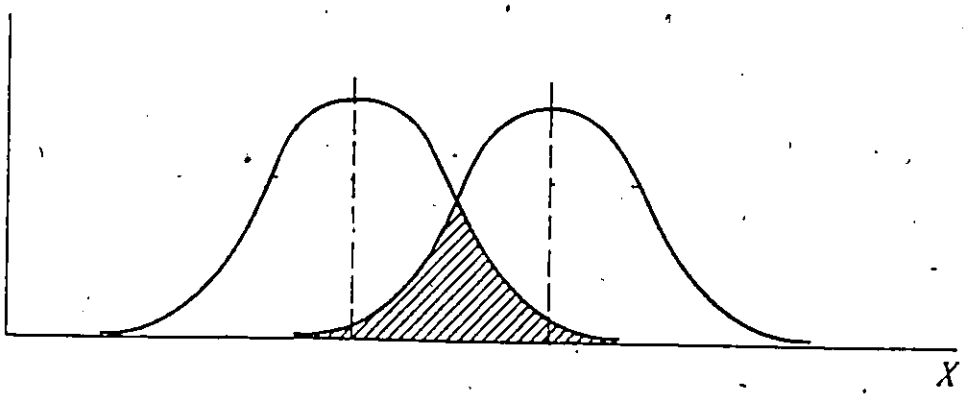

Figure 3-4
Two hypothetical classes of change overlap. The shaded area represents the amount of pixels mis-labeled by use of the maximum-likelihood decision rule (from Swain & Davis, 78).

Another example of a spectrally based technique is the Change Vector Analysis technique (Malila, 80). If { f(i,j,1) } and { f(i,j,2) } are two pictures of the same are, their distance, for instance their euclidian distance, is computed and compared to a threshold TH :

(f(i,j,k,1) - f(i,j,k,2))**2 <= TH ===> decision of No-Change

(f(i,j,k,1) - f(i,j,k,2))**2 >= TH ===> decision of Change

An angle of deviation can be calculated, giving a clue for the nature of the change (typically, by means of the scalar or vectorial product). If several pictures are used, several vectors are available. The set of vectors forms a trajectory in the feature space

which may have a pattern characteristic of the change.

3.3.4 Techniques Based Upon Spatial Analysis

These techniques are either used to assign a class of change to a pixel, or to relabel a previously labeled pixel, (Gurney, 83). In both cases, the spatial context of a picture is used to make decisions. An example of a labeling technique, is to create beforehand sets of homogeneous pixels, and then to use another technique of change detection. In this case, a pixel is labeled with its neighbourhood. An example of a relabeling technique is to consider a neighbourhood of a pixel and to apply a criterion based upon a majority: if the majority of the neighbourhood belongs to a given class, the pixel is relabeled to that class. The context of a picture is its neighbourhood but also the picture at large. For instance, a cloud can be confirmed by the detection of its shadow which may be distant. The underlying idea here is that single pixels do not change, but only groups of them. These techniques are often seen in conjunction with others (Gurney, 80,82,83), (Swain & al., 81).

3.3.5 Techniques Combining Other Types Of Data, And Knowledge.

Most of the studies about automatic change detection techniques achieved mitigated results (Tinney & al, 83), (Braun, 84). It is generally acknowledged that isolated techniques are too crude, and do not use a sufficient number of data types to provide accurate results with a low variance (Colwell & Weber, 81). The trend is thus to use other data types and even, to a limited extent, knowledge, such as in (Metzler & al., 82,83). The different types of data include (Tinney &

al., 83) the texture; the site, by use of maps giving a clue of what should be found beforehand (Tinney & al., 83); and the shape of the objects found in pictures, when pixels can be grouped with a homogeneity criterion (Shimoda & al., 83). It may also be useful to use different types of knowledge, such as forestry or agricultural practices such as used by photointerpreters (Pestre & Malila, 83). Finally, the synergy of different techniques by the creation of a decision process on top of them could significantly increase the accuracy and reliability of the results. (Woodcock, 82).

Studies which investigated the use of other types of data and knowledge are first attempts to increase the efficiency of currently existing algorithms. (Metzler & al., 82). However, they still heavily rely upon conventional approaches, though the use of "knowledge-based" systems have a strong potential (Mooneyhan, 83) (Tinney & al, 83). It may be also assumed that remote sensing traditionally uses quantitative approaches, (Swain & Davis, 78), and that embodying qualitative concepts or reasonings is still considered as hard tasks. The consequence is that the proposed qualitative decision processes are often very simple, such as a majority-based decision process in combination with different techniques (Likens & al, 82).

Shimoda & al. (83), however, propose a syntactic approach rather than the more classical decision-theoretic process. In this study, rules and grammars are employed. Several research teams are interested in pattern-recognition techniques based upon a syntactic approach, also qualified as "structural", or "linguistic" (Fu, 76), (Kettig & Landgrebe, 76), (Brayer & al.), (Pavlidis, 79), (Shimoda &

al., 83). This approach is considered as different from the "symbolic" reasoning approch of knowledge-based systems (Tinney & al, 83).

3.4. THE PROMISES OF KNOWLEDGE-BASED SYSTEMS FOR CHANGE DETECTION PURPOSES

All the improvements proposed above were intended to exploit the 'context' of a picture in its various aspects, spatial context, temporal context, other non-pictorial data, and maps. The recognized problem here is the qualitative nature of the data to use in conjunction with quantitative techniques. The hope is that the decision-making process will be enhanced by the introduction of qualitative reasoning. Studies have already been made for identifying the existing knowledge to introduce in such systems (Pestre & Malila, 83). It is believed that knowledge-based systems may increase results in change detection because of their capability for efficiently using 'context' (Mooneyhan, 83). An example of what a knowledge-based system could perform in change detection is proposed in chapter six, based on the use of a knowledge-based system, SEVERE, presented in chapter five.

CHAPTER 4

A HIERARCHY OF KNOWLEDGE-BASED SYSTEMS

4.1 CHOICE OF THE ORGANISATION MODEL

The design of a system capable of handling numerous and varied requests in remote-sensing requires building fairly different and specialized sub-systems. However, it is a characteristic of remote-sensing that these sub-systems must interact to efficiently perform their respective tasks. For example, detecting changes from Landsat pictures requires an estimate of the accuracy of the match between the available data (such as maps and pictures). It may also be necessary to incorporate results from an expert specialized in detecting cloudy regions. If we are to build a system including several experts such as the ones mentioned above, they must be linked in some manner.

The alternative strategies for designing such a system include the following :

1. To build a large expert system which would handle all the problems peculiar to remote-sensing.

2. To create an expert system with numerous sub-modules(/sub-experts), all linked to the same blackboard and all acting as slaves of the main expert. This approach was first implemented in (Reddy & al, 76), and also chosen in (Nagao & Matsuyama, 81) (cf. figure 4.1).

3. To create a network of expert systems, each node being an expert specialized for a given task: detecting changes, quantifying the geometrical characteristics of Landsat pictures and maps, etc...

The first alternative would need to handle the emulation of scores of conventional programs currently used in remote-sensing. A huge knowledge-base, containing knowledge about fairly disparate aspects of remote-sensing is needed. Clearly, the problem at hand, by its multiple aspects, completely differs from the ones solved by expert systems such as Mycin (Buchanan & Shortliffe, 84) or Prospector (Duda & al, 78). Such a design choice would be against the essence of expert systems which handle very specific and delimited domains. Critical issues in the design of expert systems are now well understood. One of them is the coherence, or consistency of the knowledge manipulated by an expert system, and in this regards, problems caused by large volumes of knowledge are still unsolved. The classical approach for preventing these problems is to split a large knowledge base into smaller independent sub-sets, the underlying strategy of the last two alternatives. Another issue is the controllability of an expert system. A search by use of a large knowledge base may become unrealistic whenever an expert system has to realize a long series of tasks before finding an answer, or when too many data are available, or when too many tasks may be performed at the same time. When such situations arise, it becomes very hard for the knowledge engineer to check whether the system selects the right tasks in the right order.

The second alternative can be seen as a special case of the third one : it is a network implemented in a tree fashion with only two levels. Such an approach was chosen in (Nagao & Matsuyama, 81) for the implementation of an expert interpreter for aerial photography. In this case, all the sub-systems share a common blackboard, and they are triggered by matching the content of the blackboard with their

knowledge-base elements. The goal of this expert was well defined and the system did not have to handle other problems in remote-sensing, such as geometric corrections, cloud detection which were considered as solved by conventional programs. Unfortunately, this is not the case for Landsat imagery and knowledge, hard to embody by conventional programming, is required to achieve good results (Gurney, 82). Moreover, the system was complete and no extension was forecast. Thus a two-level organization was sufficient in this context. It is difficult to know beforehand the number of experts which will be required, though it will be undoubtedly large, possibly too large for a single blackboard to stay the only medium for communications between experts. Furthermore, the amount of communication may vary greatly from expert to expert. If we want a flexible architecture, capable of handling different problems and capable of evolution, we must reject the second approach.

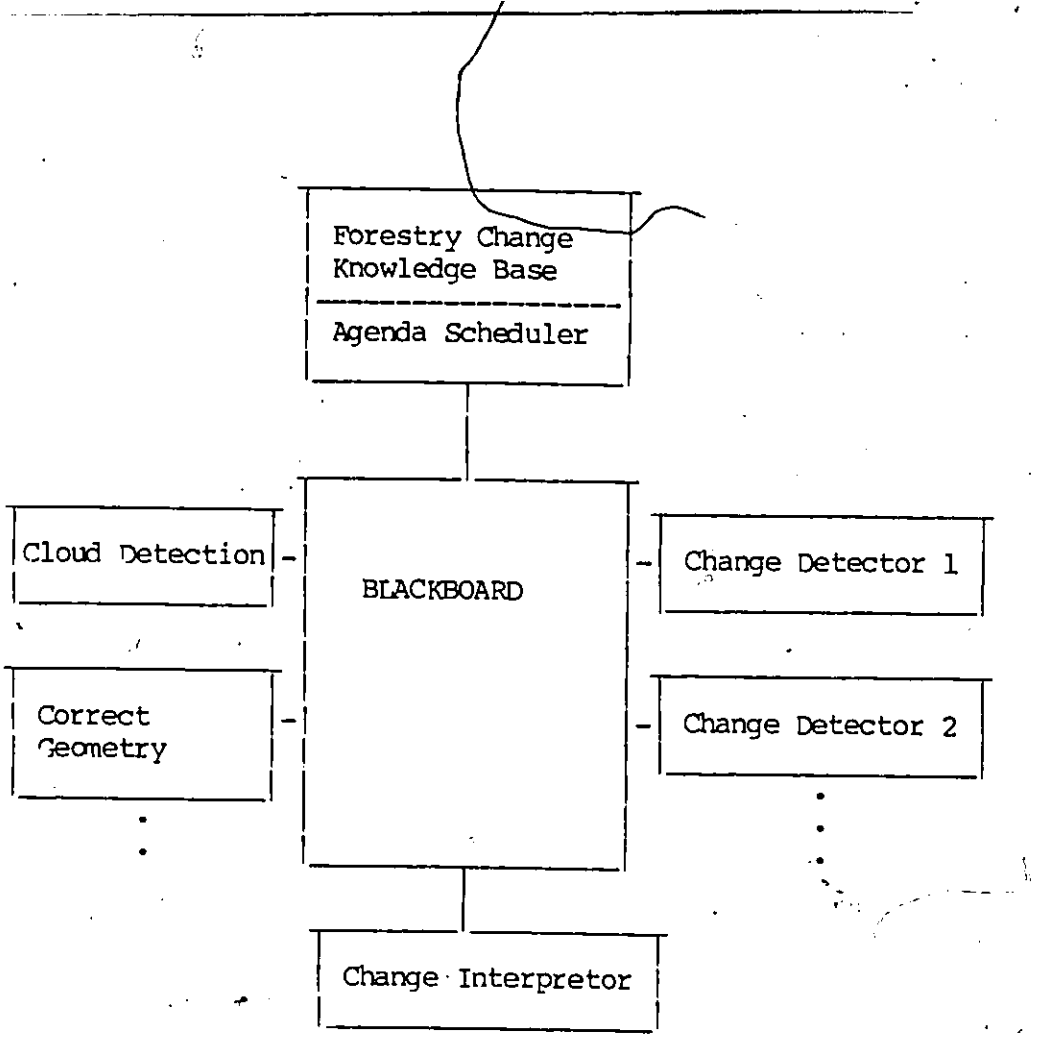

Figure 4.1
Example of the organisation inspired from (Nagao and Matsuyama,81) for a system dedicated to detecting changes in a series of Landsat picture. The blackboard structure is chosen enabling cooperation between specialized experts

The network approach does not have the drawbacks of the first two. The penalty paid, however, is that control aspect of the overall network becomes more complex. Controlling a network can be performed through either ordering the network nodes, leading to a hierarchical network or a tree structure, or through a possibly more complex and more general non-ordered network. In the hierarchal approach, the interactions between nodes are well defined. The gain is in clarity in the functioning of the system and a simpler implementation, the

loss is in the flexibility of the system, since the links are now "oriented".

The hierarchical approach was preferred, both to ensure clarity in the design approach and in the implementations, and because such a structure can be seen as a particular organization of the network, thus it may be altered in the future to become a genuine unoriented and unordered network. Clarity will be of help to keep the overall system consistent and manageable (either for debugging or for future expansion). In any case, it is likely that the overall system will need some sort of ordering. For instance, a node dedicated to the detection of a particular type of change will naturally be controlled by a "manager-node" which would study the general problem of detecting changes in general.

4.2 DESCRIPTION OF THE PRESENT APPROACH

The top of the tree would typically be a knowledge-based system used as an interface between the user and the rest of the nodes. It should interpret the user's request and set broad goals for the systems immediately below it. Each sub-system of the tree is called a node. The top node is called the root and the terminal nodes (the ones with no node below them) are called leaf-nodes (Figure 4.2).

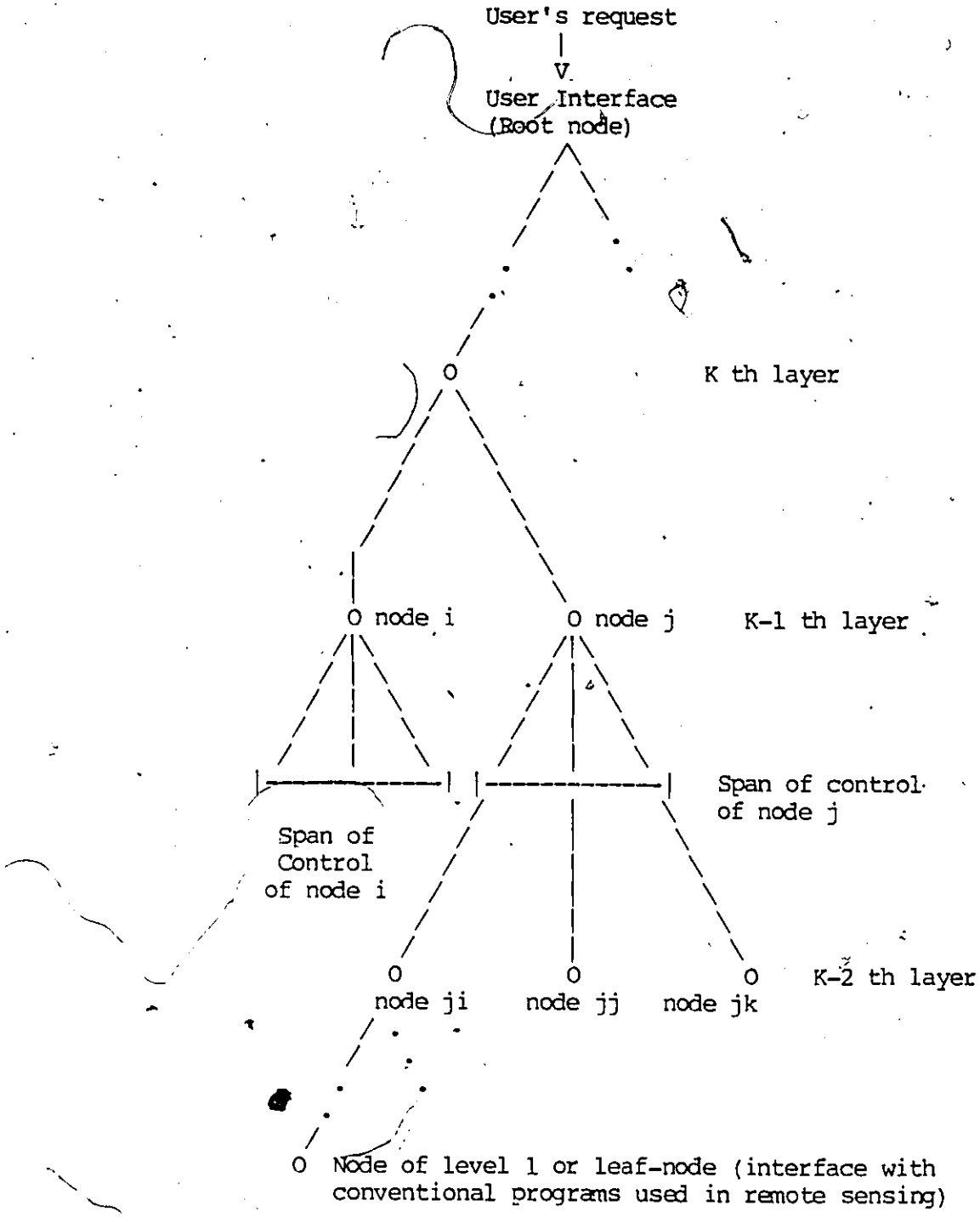

Figure 4.2
General organisation of the proposed system of experts for remote-sensing. Each node represents an elementary and specialized expert system built from the same shell (apart from the root and leaf-nodes)

The hierarchy implies that a node is the manager of the nodes immediately below it. It also implies that each node executes the commands sent by its manager (the root has the user for manager). A user's request is then interpreted and restated as sub-goals, each sub-goal being itself restated and passed to the next lower level. Eventually, productive work, in distinction to intellectual work, must be performed by the lowest level in the hierarchy. The leaf-nodes have then a special duty: to activate already existing algorithms and to return solutions in a usable form. From this analysis, it becomes clear that the root and the leaves are specialized knowledge-based systems. The presented shell was designed to help in the instantiation of the non-leaf nodes. The overall system is thus a tree of nodes communicating through interfaces.

As explained in the following chapter, each shell contains a blackboard and an interface used to transfer commands and/or data between its blackboard and the blackboard of the other experts with it may communicate (Figure 4.3).

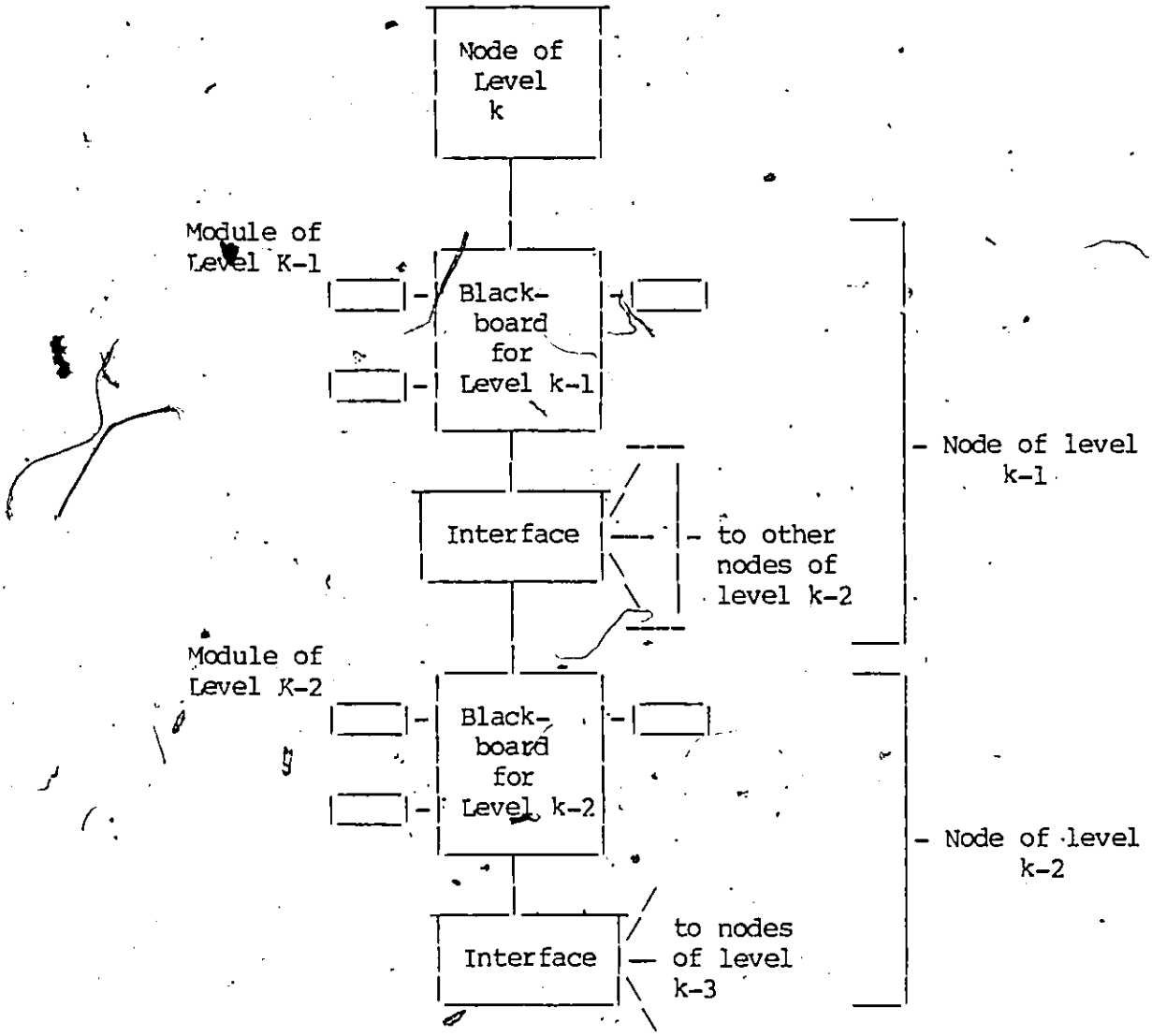

Figure 4.3

A hierarchy of knowledge-based systems designed by use of the general blackboard approach.
Each system has an interface the duty of which is to transfer requests and/or data between its blackboard and those of related systems.

4.3 INTERACTION WITH THE SHELL ENVIRONMENT

The overall system can be seen as interacting with the outside by a number of different interfaces :

1. The root node, by interfacing with the end-user. Through that node, the user may give a request with some initial data, and the expert system its answer.

2. The knowledge acquisition system, by interfacing with human experts in remote-sensing. The knowledge-base of each node may be accessed by the human expert.

3. Each node has the capability of reading data from the outside via a part of its interface called the data-handler. Thus, for example, information from an external geographic data-base can be incorporated into the blackboard as needed.

4. The explanation facility, by interfacing with the end-user or experts in remote-sensing.

5. The leaf-nodes, by interfacing with procedurally-based remote-sensing software. However, the procedurally- based software should be seen as an extension of the overall system.

4.4 SUMMARY

The present chapter showed an organization allowing expert systems to cooperate on a common problem, namely, the one entered at the root-node. We also saw in chapter three how distributed expert-systems could be of interest in remote sensing. The following chapter will thus present a system which will allow the building of a hierarchy of expert systems for remote sensing. An example of an implemented hierarchy of experts with two levels is presented in

chapter six.

CHAPTER 5

PRESENTATION OF SEVERE

SEVERE is a Prolog-based tool for developing hierarchically organized expert systems and a rule-based expert system for classification problems. It is composed of a shell (section '5.1'), an explanation facility (section '5.2'), a knowledge acquisition system (section '5.3'). The nodes of the hierarchy presented in chapter four are created by instantiating SEVERE's shell. The main characteristics of SEVERE are summarized in section '5.4'.

5.1 THE SHELL

5.1.1 Overview

The architecture of the expert system shell, represented in Figure 5.1, is composed of the following elements :

1. A "blackboard" (and its related interface) for the storage of any information relevant to the problem such as data, (intermediary-)results, a list of previously decided actions to perform (the "agenda"), and generally speaking, any dynamic information the value of which may change during the consultation.

2. A "base" for storing "knowledge". The system exploits two kinds of knowledge: the object-level and the meta-level. The former deals with the problem at hand, and is contained in the "object-level knowledge-base". The latter, the meta-level knowledge base, deals with methods for arriving at results, a process called "deriving strategies".

3. An "interpreter" the duty of which is to apply the available knowledge. In the context of a knowledge representation based upon production systems, its task is to "fire" rules. Here too, we make the distinction, between the meta-interpreter and the object-level interpreter.

4. An "arbitrator" or equivalently "consistency enforcer" which chooses the "best" solution for the current problem, among possibly several concurrent and contradictory hypotheses. Two arbitrators should exist: the meta-level and the object-level arbitrator. However, since the meta-level finds only one result, the strategy, when called, there is no need for a meta-arbitrator.

5. A justifier-interface which keeps track of the way the (intermediary-)results are deduced. The information thus stored can be read by the explanation facility, to provide justifications about the way the system makes deductions.

6. A "scheduler" which manages the different elements. Each of the other modules are under the control of the scheduler which executes "strategies" proposed by the meta-level.

7. The elements presented above are "classical components" of an expert system shell. For the purpose of a hierarchical system, we must add an interface for the transfer of commands and/or data between nodes(/shells). The interface has as a "front-end" a formatter, which may transform the final result into a form understandable by the manager-node .

The main elements and their roles are summarized in Table 4.1.

Name of the Module	Purpose of the Modules
Blackboard	Storage of Data and (intermediary,final-)results
Blackboard-interface	Interface between all the existing modules of the shell and the Blackb.
Knowledge Base	Storage of the Knowledge relevant to problem
Meta-Knowledge Base	Storage of the Knowledge about the deduction process
Meta-Interpreter	Module applying the available meta-level knowledge
Object-Interpreter	Module applying the available object-level knowledge
I/O Interface & data handler	Exchange of information between the nodes
Arbitrator	Resolution of the possibly conflicting hypotheses.
Justifier Interface	Storing of traces about data and results for later exploitation by the Justifier
Scheduler	Shell manager

Table 5.1

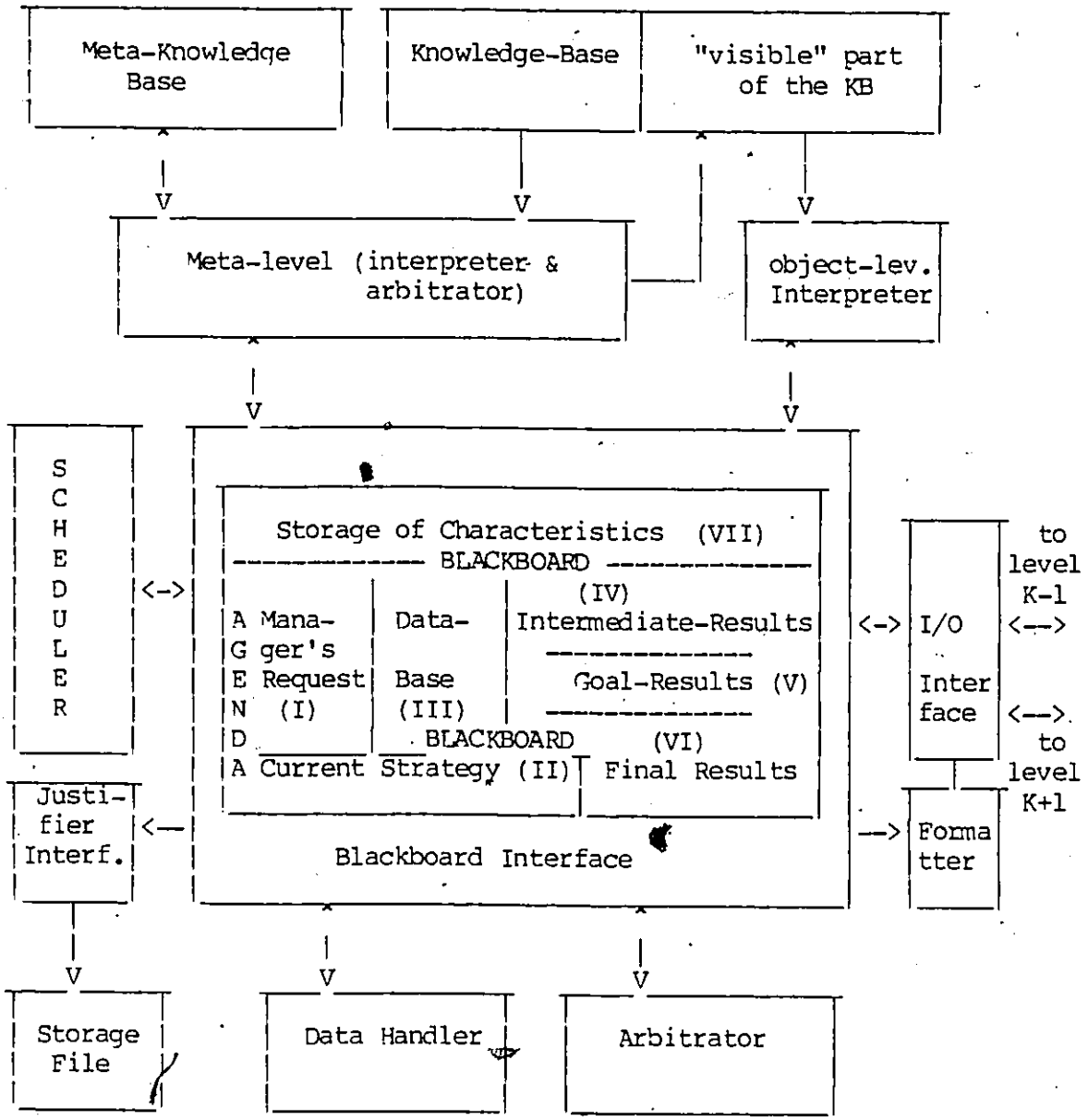

Figure 5.1

Overall Architecture of the Consultation Part of the Shell
(the arrows represent data and/or control flows)

5.1.2 : The Blackboard

The duty of a blackboard is to store any kind of relevant information for the problem at hand. Several means of storing data have been proposed such as the context tree (Buchanan & Shortliffe, 84), or the semantic net (Duda & al., 78), but the blackboard approach is recognized as the most general one. A change in the instantiation of the present shell, will change the instantiated part of the present blackboard, but not the blackboard itself (mainly the set of goals, of sub-goals, and the characteristic of these elements, such as whether they have to be proved several times or only once). The context tree of Emycin has to be completely instantiated and a tool is provided to alter its structure. Clearly, a change in the instantiation changes the structure itself. In Prospector (Duda & al., 78), where models are used to embody expert knowledge, the semantic net approach was preferred. In this case, the semantic network was found quite adequate to match data and (knowledge-)models. It is also generally recognized that production systems and blackboards are two complementary tools.

As shown in figure 5.1, the blackboard is divided into seven sections (the capital Roman numerals refer to Figure 5.1) :

1. Two sections store the actions to perform. The first (I) stores the manager's request, and the second (II) the strategy designed by the meta-level.

2. Four sections are dedicated to the storage of information relevant to the problem. Section III stores the information available at the beginning of the consultation; section IV is reserved for the intermediate results; that is, those results which are not a part of the direct solution of the current problem, but may help for its solution; section V contains the hypotheses or solutions obtained by the application of the meta-level and the object-level knowledge. Since several paths in the knowledge base may confirm or invalidate a solution or may deduce possibly contradictory solutions, the arbitrator will have to find the "best" solution which is then stored in section VI. This last solution is returned to the manager, possibly after a translation by the "formatter" module.

3. The last section (VIII) is reserved for the other data which are not directly useful to finding a solution but may help the system to make deductions with a better efficiency. This section, which may be tailored by the knowledge engineer, already contains several predicates which represent characteristics of the elements present in the blackboard. For example, one predicate stores the list of the elements which must be confirmed, and thus proved several times, if used in the deduction process. Depending upon the problem at hand, certain goals or sub-goals are either considered as acquired once found, and thus there is no need to study them again, whereas others may be questioned, and thus are to be proved again. The system will typically try to confirm or invalidate an element of the second type by exhaustively

trying to find the paths concluding about it in the knowledge base. Another predicate stores the results already sent to the manager, to prevent repetitions in the answers, while a third records the meta-rules already fired, in case the meta-level decides to put some of them back into the meta-knowledge base. These data are important for the node, to function efficiently, but do not contribute directly to the search of a solution.

To highlight the different roles of the blackboard, we present an example. We make the following assumptions:

1. Some node, called N, has the following set of goals: {a,b,c}

2. A request is received from the manager-node of N "to find the best solution".

3. The node N has been already activated once and "a" was returned to its manager at that occasion.

4. Finally, the knowledge-base of node N contains the following rules:

   b' —> b

   b''—> b'

   and b'' is given, i.e. is stored in section III of the blackboard of node N. b'' may be known from a previous activation of the node or may have been transferred into the blackboard by its manager.

In executing the request the following steps are performed which alter the blackboard of node N as follows:

i) Using the first rule, b' is deduced. Since b' is not a goal, it is stored in section IV.

ii) Using the second rule, b is deduced. As b is a goal, it is stored in section V.

iii) When the arbitrator is called, section V contains a and b. The arbitrator will choose b as the final solution since a was previously sent, so b is stored in section VI.

iv) The goal b is stored as an argument of the predicate "already-sent-results" of section VII.

5.1.3 The Blackboard Interface

This module contains all the procedures useful for the following operations:

1. Deleting data from the blackboard,
2. Finding data in the blackboard,
3. Adding data to the blackboard,
4. Resetting the blackboard, at the beginning of the consultation session.

A module can interact only through the interface, thus ensuring flexibility of the blackboard. The advantages of having an interface are clear. Further changes in the blackboard structure itself will imply changes in the interface and not in the rest of the shell or in the way knowledge should be written. For example, the question "is X in the blackboard ?" will not change in its formulation and its answer will be provided by the interface definition "find-in-bb/2" the

content of which may change when the blackboard structure is changed. However, the designed blackboard is simplistic in its realizations.. For example, SEVERE has no embedded feature allowing it to store geo-coded data such as maps and pictures. Implementing this capability is beyond the scope of the study. In fact, this is highly dependant upon the host-computer, and upon a number of technical problems ("how to store matrices in Prolog", "how to store pictures coded in vector format", and so on).

5.1.4 The Object-level-knowledge-base

A knowledge base is classically composed of a set of facts, a set of rules and a stopping condition. Facts can be seen as rules with no premise and thus will be stored in the same way as rules. A rule is composed of a set of conclusions (the action part), a set of conditions and elements whose existence are to be verified (the situation part), a certainty factor (presented in '2.6.3' and justified in '5.1.7'), and a rule number. All the rules have the same general format, with each set being represented by a list of elements.

There is no predefined format for an element, each element must to be defined by the knowledge engineer. A sample format is presented in the instantiations (see chapter six and the appendix). Since Prolog extensively uses pattern-matching, no constraint is needed on the format of the elements, such as being a four-tuple as in Mycin (Buchanan & Shortliffe, 84). Two elements are alike if they can be matched by Prolog, whatever the selected format. One of the obvious advantages of such an approach is that the format may vary for different classes of elements manipulated by the same expert system without having to define the meaning of a format to the system.

The stopping condition takes on one of two forms: if the element does not have to be proved again once found, then the object-level interpreter will stop the search; or if the element has to be proved again, then the object-level interpreter stops when no visible rule are firable. The stopping condition is actually embedded in the interpreter.

The "(potential-)knowledge-base" comprises all the rules available to the node. It is read and manipulated by the meta-level. A part of the potential knowledge-base is selected and made "visible" by the meta-level for the object-level interpreter. Selecting a part of the knowledge and making it visible is usually the result of pruning and sorting the knowledge-base (in many expert systems). This is also the case here and a tool is provided for this purpose : a pattern matcher (see '5.1.11').

5.1.5 The Meta-level

The meta-level is systematically consulted before any cycle of search in the object-level knowledge-base (as exemplified in the presentation of the operating mode in '5.1.13'). The meta-level has access to a meta-knowledge base containing rules which typically draw conclusions about actions to be performed by the scheduler. These actions may be the activation of a peculiar module, such as the arbitrator, or the activation of a sub-node, of which it is the manager. It may also perform some sorting and pruning of the object-level knowledge base. The situation-part of the meta-rules are a peculiar description of the blackboard content. By firing meta-rules, the meta-level creates a "strategy" i.e. a n-tuple of actions to perform. A stratregy is created after each consultation of

the meta-level. All these reasons highlight the fact that the meta-level is often considered as an expert system itself.

A meta-knowlege base contains meta-facts, meta-rules and a stopping condition. As for the object-level knowledge base, meta-facts can be seen as particular meta-rules and have the same format. The format of a meta-rule is different from the one described for the object-rules. A meta-rule is formed by :

1. a set of actions; those actions will be executed under the control of the scheduler if the rule is fired.
2. a set of conditions; Each condition typically represents a description of the blackboard to verify.
3. a priority number; The priority number helps the meta-interpreter to sort the rules to execute.
4. a rule number.

The remarks formulated about the format of the object-level elements are also valid for the meta-level elements. However, if the meaning of an object-level element is left to the user, the meta-level elements are procedures, explicitly defined by the knowledge engineer. Each meta-level element has a definition in Prolog stored in a file and consulted by the node at the beginning of a session.

The stopping condition is a failure to find a firable rule, either because no existing rule can be fired, or because no more rules exist. The latter case may happen after deletions performed by the meta-level itself. Since no controler of the meta-level exists, the meta-interpreter has access to all the rules stored in the meta-knowledge base.

Historically, the meta-level was designed to contain only rules for sorting and pruning the object-level knowledge-base. Thus, actions to be performed, such as the activation of a module or a node, would be decided by the firing of an object-level rule. However, in the present system, the manager node can be considered as part of the meta-level of its sub-node because the request sent to the sub-node may involve a pruning and/or a sorting of its knowledge-base or, more generally, may direct the sub-node to a particular search. The same remark also holds for the action of calling a sub-node: it is a "meta-action" since it privileges the knowledge-base of the invoked sub-node as compared to the sub-nodes. For these reasons, the activation of a module or a sub-node is left to a meta-rule.

Moreover, since the meta-actions are essentially procedural and designed for very specialized reasons, it is reasonable to attribute them to the meta-level, instead of grouping them with object-level rules, which typically deduce a diagnosis. Thus, we can differentiate the meta-level which is procedural and the object-level the purpose of which is to deduce or provide a diagnosis. This differentiation keeps valid the formal definition of the object- and meta- levels (Davis & Lenat, 82):

To deduce a diagnosis from ad-hoc rules (object-level),

and to provide a way to reach a solution in the search process of the object-level (meta-level).

There exists no meta-meta level, i.e. another level, guiding the choice of the meta-rules to fire, and of the actions to perform. However, certain design choices must be made at this level. A meta-rule is automatically deleted from the meta-knowledge base once fired since it is likely that the rule will not be of use anymore, and since we want the meta-level not to repeat the same strategies. Such a decision belongs to the meta-meta-level. We also point out that meta-actions are not limited in their duty and thus they can alter the meta-knowledge base itself. For example, it may be desirable that meta-rules be deleted, that previously deleted meta-rules be stored back in the meta-knowledge base, or that their priority be changed. Such actions typically belong to the meta-meta level (since they sort or prune the meta-knowledge base).

The meta-interpreter uses priority numbers (chosen from 1 to 20, an arbitrary interval) assigned to each rule to sort the meta-rules. The sequence of tasks performed by the meta-interpreter is as follows:

1. The current priority level is set to 1.

2. The meta-interpreter begins to evaluate the premise of the rules of current priority level N.

3. If one is firable then,

    i) All the rules with priority N are scanned by the interpreter;

ii) The conclusions of each fired rule is appended to the previously found conclusions;

iii) The interpreter stores a trace of the firing for future explanations; and

iv) The interpreter stops when no more rules of priority N can be fired. The ordered set of conclusions to execute represents, by definition, a newly created strategy.

If the interpreter cannot fire any rule of the given priority then

i) If N <> 20, then N = N + 1, and a new cycle is performed, starting from step 2, or

ii) If N = 20 then the meta-level fails to create a strategy, a situation which corresponds to the end of a consultation session of the node (see '5.1.13').

It must be noted that, once a priority is set, the meta-knowledge base is scanned only once. This is justified by the fact that the premise will typically be a description of the situation (i.e. of the blackboard) and that the firing of a meta-rule will not change the immediate situation. It is only when a strategy is chosen that the object-level is activated. Thus, another scanning of the meta-knowledge base would not find any new rule to fire. We shall see below how the object-level interpreter differs in this respect.

There actually exists two ways of ordering the rules and thus the actions to perform within a strategy. The first one is merely linked to the existence of the priority number. The second one corresponds to the natural ordering of the rules inside the meta-knowledge base.There is reason to believe that the double ordering capacity will answer most problems of instantiation of the shell.

The meta-interpreter is typically one of type C. The interpreter scans the meta-knowledge base and fires the first firable rule, the order being given by the ordering of the rules and the priorities. However, since there exists some form of meta-meta-level, the system is not a pure C-type engine. For example, by use of adequate meta-actions, the interpreter may alter the meta-knowledge base and thus induce some type of action at the following consultation of the meta-knowledge base. But it should be noted that the example represents a non-natural way of using the meta-level and this feature should be used with care.

The firing of rules depends heavily upon the rules and data-structure. The design of an interpreter must take these into account. In Emycin, the rule format at the object and meta-level is the same, allowing the design of only one interpreter. However, meta and object rules are very distinct in nature and meaning for the purpose of the shell presented. Activating a sub-node for example, is a task highly procedural in nature and is an "action" to perform. On the other hand, a conclusion at the object-level will typically be a decision, a diagnosis. No action is to be performed, apart from storing the conclusion in the blackboard.

Another consideration is the way interpreters should function. It is posible that object level interpreters do not function always in the same way. This may depend upon the instantiation. Having two independant object and meta-intrepreters allows the tailoring (for optimization) of the object-level interpreter. Two instantiations of the shell are given in chapter six and highlight the differences between the two object-level interpreters.

5.1.6 The Scheduler

When the overall system focuses on a particular node, the first activated module is its scheduler. Its purpose is to execute the actions decided by the meta-level. Its main task is to execute a series of cycles, one cycle containing the two steps:

1. Consulting the meta-level,
2. Executing the actions selected by the meta-level and stopping whenever consulting the meta-level fails.

The stopping of the scheduler corresponds to the end of the consultation session for the currently activated node. Its central role is highlighted in the operating mode section (see '5.1.13').

5.1.7 The (Object-level) Interpreter

The interpreter fires rules made available by the meta-level. There are two ways of firing rules, in backward chaining and in forward chaining; both are implemented in the shell. The choice is made via a parameter set when instantiating the shell. This parameter may be changed by the meta-level, allowing more flexibilty in the deduction

process. To prevent a second firing of a rule in the deduction process, the interpreter keeps track of the fired rules. Each conclusion of a fired rule is evaluated and a trace of the firing is kept for use by the explanation facility.

As already mentioned for the meta-interpreter, the object-level interpreter may need some tailoring depending upon the task to perform and the way the elements are coded. What is part of the shell is the firing of one rule, whatever the mode of chaining. The nature of the tailoring to perform depends upon the particular problem at hand, and an example is presented in chapter six and the appendix. Having to tailor the interpreter is in fact an efficient way of dealing with the potential insufficiencies a general shell may contain. Experience will induce whether a more sophisticated shell can be designed to encompass various types of interpreters.

The object-level-interpreter executes the following two steps :

1. The interpreter scans the available knowledge-base and performs the following operations for each rule:

   The premise of the current rule of study is first evaluated:

   If the rule is firable then the conclusions of the fired rule is stored in the relevant part of the blackboard. If the conclusion is a goal, it is stored in section V (Figure 5.1) of the blackboard. If the conclusion is not a goal, it is stored in section IV (Figure 5.1) of the blackboard. Before the transfer to the blackboard, the interpreter updates the confidence which was previously given to each of

the conclusions (see below). Then, the interpreter stores a trace of the firing for use by the explanation facility.

If the rule is not firable, the interpreter studies the next rule present in the work space.

2. Once all the knowledge-base is studied, the interpreter checks whether any rule has been fired.

If not, the interpreter stops, and the scheduler continues.

If something was deduced, then the interpreter checks whether all the goal-set was deduced.

If so and if all the found goals don't need to be proved again (to be deduced again by another path in the knowledge-base), then the interpreter stops and the scheduler continues.

If all the goals have not been deduced, or if some of them have to be proved again, then another scanning of the knowledge-base is performed, since the newly deduced elements may allow the firing of previously unfirable rules.

At present, uncertainty is handled in the same way as in the Mycin expert system. Each rule is assigned a "certainty factor", a number between -1 and 1 (see APPENDIX A). Mycin's theory was chosen because it has been extensively tested. One of the biggest drawback of Mycin's theory is its experimental nature. The outcome of its implementation can only be validated by experts and their intuitive feeling. The choice should thus be subject to extensive testings.

Other methods, implementing more theoritecal approaches may easily replace the present choice, since uncertainty is handled by only one module in the shell.

5.1.8 The Arbitrator (or Consistency Enforcer)

At the meta-level, no arbitrator exists since the action parts of all the fired meta-rule are concatenated into one list of actions to perform so there is no actual selection among the meta-actions. This list represents the strategy created by the meta-level. The only conflict which may arise lies in the nature of the actions to perform. This is a semantic problem to be solved by the human expert, rather than by an arbitrator, by appropriate use of the priority numbers and the ordering of the rules within the knowledge-base.

At the object-level, since the deductions may be contradictory, the arbitrator must select, for instance, the best solution among the set of competing solutions It should be able to recognize whether a result is valid or not, whether an anomaly exists, whether results are complementary or incompatible, and should propose the best interpretation (which may be the absence of intrepretation in the case of doubt). By anomaly, we mean the existence of two or more results which should not exist at the same time. Such a situation arises when the object-knowledge base is incomplete or when the scope of the expert system does not include the current problem. These are fundamental problems in the design of expert systems. The arbitrator finds the best (sub-)goal(s) among a set of competing elements, and has elementary features to help the knowledge engineer define when a result is valid and when an anomaly arises. The features are useful

safeguards to be employed during the instantiation process.

We showed how the meta-level and object-level interpreter may have to be tailored to the problem at hand. This may influence the arbitrator and imply a related tailoring. Chapter six and the appendix present an example of a tailored arbitrator, and highlight the existing similarity between the tailoring performed for the object-level interpreter and the one for the arbitrator.

5.1.9 The Justifier Interface

Since the justifier and the shell are distinct, a justifier interface exists for each node, which stores in a file (one per session), for later consultation by the justifier, all the relevant information concerning the conclusions drawn at the meta and object level. For each conclusion, the object-level stores a list consisting of:

1. The conclusion itself.
2. A strategy label designating the strategy under which the conclusion is found.
3. The rule used to draw this conclusion.
4. The measures of belief and of disbelief associated with the conclusion.

In the same way, the meta-interpreter stores the actions concluded by the meta-level with a list consisting of the meta-rule which concluded the action. Both meta-actions and object-level conclusions are stored in the same way for the sake of a uniform processing by the justifier.

5.1.10 The Inter-node Interface And The Data-handler

The hierarchical architecture of the total system implies that interfaces exist between nodes and confer a regulating role to the interfaces. The interface is divided into sub-parts. One is the formatter and is used to translate, if needed, the final result of the shell(/sub-node) in a meaningful form for the manager. The second, called the "inter-node manager", has access to the blackboards of the shells it links. The actual transfer of commands and data is performed by this part of the interface. The transfers are peformed in a very general manner. Either a file is read and its content stored in a blackboard, or a list of command is directly transferred. In both cases, there is no semantic checking of the transferred elements.

Since the nodes are particular instantiations of the same shell, the set of the elements which may be manipulated by the Prolog interpreter must be regularly changed. This happens when the focus of interest changes nodes. In this case, it is the task of the interface to "deactivate" the node being left, and to "activate" the new node of interest. At the same time, if the focus of attention is switched from a node to its manager, the interface must store the final result obtained by the node in the blackboard of its manager (in the "intermediate results" part). If the focus of attention is switched from a node to a sub-node, it is the request provided by the manager which is transferred to its sub-node.

For simplicity and efficiency, the inter-node interface is specialized in transferring requests and short answers. If picture or geo-coded data have to be transferred into a node, it is easier to send a command via the inter-node interface telling what to transfer, and possibly, how to transfer (from where, with which protocol...), and let the data-handler of the receiving node execute the command. The duty of the data-handler is thus to interface a node with anything outside the hierarchical system. The outside could typically be a geographic information system. Such a feature is essential if we want to limit the amount of overhead in a hierarchical organization. It is also essential if we want the system to use data available in outside data-bases.

The data-handler is designed to transfer data from the outside into the blackboard of the shell. However, it may be useful in the future to increase its capability by allowing the transfer of knowledge or the altering of parameters. In fact, a whole range of implementations may be useful for this module, depending upon the task it would have to perform. It may be a simple and automatic reader of files, as in the present case, but it may become an genuine expert system: deciding what data to fetch, where to find the data, and possibly, how to translate these data in an understandable form for the shell. In the last case, it is likely that each node would be linked to the same data-handler, which would be the interface between the overall system and the outside.

5.1.11 The Pattern Matcher

The pattern matcher is used to select object-rules according to "patterns" in the premise or the action-part of the rules, which match a given list of elements. If selected, the rules are made "visible" to the object-level interpreter by changing their predicate name.

5.1.12 Files Which Are Part Of The Shell

The shell is empty at the beginning but knows where to look for knowledge, information, or data. By consulting a number of files, the shell is filled with all that is necessary to actually "instantiate" it. For the shell itself, without any constraint from the hierarchical organisation, the following must exist and be consulted by the shell:

1. The meta-knowledge base containing the meta-rules and the number of rules.

2. The knowledge-base containing the object-rules and the number of rules.

3. The emptied version of the blackboard, which is consulted each time the shell has to reset itself.

4. The set of concepts. The elements found in the meta-rules and their meaning (i.e. their procedural definition) which is defined by the knowledge engineer when instantiating a shell. One meta-element is defined and provided with the shell. This concept makes visible to the object-level interpreter some of the object-rules. When a strategy

contains this meta-element, the scheduler executes it by consulting its meaning in a file. This meta-element selects object-level rules by matching them with one of its arguments and alter them in such a way that they become "visible" to the object-level interpreter.

5. The set of conditions. The conditions are elements, found in object-rules, which have to be verified for the rule to be fired but which don't have any influence in the computation of the confidence measures of the conclusions of the fired rule. These elements are also defined by a procedure.

6. The part of the blackboard which contains information about the instantiation (the definition of the set of goals, the current mode of chaining, for example).

Once a shell is instantiated and a new node is created in the hierarchical organisation, several other files have to be consulted or are created during a session :

1. One file which is consulted by the data-handler at the beginning of any session. It may contain data the manager of the node wants to transfer to its sub-node, or data from an outside system.

2. If a sub-node is called, its answer is available in a file to be consulted by the manager.

3. The file created by the justifier interface.

4. The file containing the manager's request and which is read by the node at the very beginning of any session.

5.1.13 Operating Mode

Once a node is activated for a task, all the presented modules will be used in turn, under the control of the scheduler. After the sending of a request from the manager, the following is performed at the invoked node :

1. The interface between the two nodes receives the request, deactivates the manager's shell and activates the node of interest. If the node has been previously called, the interface restores the node back to the previous state. The interface then leaves the priority to the scheduler of the activated node.

2. The scheduler first looks at the request. Typically, there is a flag for a reset command in the request.

   If the flag is "reset", then the problem at hand is considered as new, and the node is reset by emptying the blackboard, and the work space.

   If the flag is "noreset", the request is a new question about an old problem, and the node resumes its activities from where it stopped.

More generally, by combining the capabilities of the data-handler and the internode-interface, an activated node

may start from any state. This is accomplished by resetting the node to a starting state, and via the data-handler, putting data into the blackboard.

3. The scheduler stores the request in the blackboard and the data-handler consults a file through which the manager-node transfers data.

4. The scheduler then performs a series of cycles, each cycle being defined by two actions :

1) The meta-level is consulted for a strategy.

2-a) If a strategy is proposed, it is stored in the blackboard, and is executed by the scheduler. These actions may be to request something from a sub-node, to request a module of the node (the interpreter, the formatter, etc) to perform an action, or to perform a complex procedure involving the use of several modules or subnodes.

2-b) If the meta-level fails to propose a strategy, the scheduler ends the session and priority is then given to the interface, which deactivates the node, (stores certain data belonging to the node), reactivates the manager node, and transfers the final result found by the sub-node into the blackboard of the manager.

Once the strategy is executed, the scheduler begins another cycle (until state 2-b is encountered). It is worth noting that the interpreter and the arbitrator are called by the execution of an action within a strategy. This enhances

also the flexibility of the system since these modules may be called whenever it is desirable, instead of at fixed times. For example, the arbitrator may be requested, at any time, to find the best element among a list.

5.1.14 Design Choices For The Inference Engine And The Overall System

We use the terminlogy presented in (Laurent, 84) and also in '2.5' to characterize the inference engine. The engine of the shell is of the O-A type, like Mycin. The selected object for processing is always the last generated goal-object in backward chaining. Thus the selection of the action is the first firable rule which includes the goal-object in its conclusion part. The selected objects for processing are always the elements already present in the blackboard (the "known facts") in forward chaining.

Many expert systems have an engine of the C-type. For example, an expert system whose inference engines selects and fires the first firable rule encountered while scanning the knowledge base belongs to that type. In this case, the object to process (the conclusion) and the rule to fire are chosen at the same time. An engine of type C seems intuitively less "universal" then one of type O-A or A-O. Choosing an object (respectively an action) may come from criteria which are specific to the set of objects (actions) at hand. The set of criteria which may choose the couple (object,action) at one get may not include the concatenation of the criteria about the objects and the criteria about the actions. To highlight the advantage of an O-A type engine over a C-type engine, we present a particular aspect of

the problem of detecting changes in remote-sensing. Suppose we want to know whether a given region may be covered with clouds in a given picture. A request is sent to the relevant expert which, by its meta-level, would select the rules concluding about the existence of clouds. Thus, by pruning the object-level knowledge base, the object to process is chosen. Then, the meta-level may also sort the remaining available rules by use of criteria, such as, "the rules having the highest confidence measure should be studied first", and so on. The example clearly shows two points:

1. A C-type engine cannot handle such specialized request since it cannot select what is to be searched, and since it cannot sort the rules.

2. The meta-level plays a major role in the selection of the objects to process and the rules to fire. In fact, such flexibility can only be achieved by use of a meta-level.

The choice between O-A and A-O is rather arbitrary. It is however generally recognized that criteria about objects are easier to find and more determining than the criteria about actions. This explains why in practise, the chosen type is always O-A. In fact, only one expert system, dedicated to playing chess, of the A-O type can be found in the litterature (Wilkins, 82).

Since the shell was designed to create nodes which should remain simple and handle simple tasks, the complexity of the overall system should come from the multiplication of nodes and not from the sophistication of its nodes. In particular, the knowledge base of each node should remain "small", i.e., the particular problems

associated with handling large knowledge bases, rules consistency, search space, etc... should not occur.

All the presented reasons are in favor of an O-A type engine, and can be summarized by the fact that a general shell should be "robust" in order to handle many different applications. Robustness is favoured by the proposed approach over a more sophisticated and perhaps unjustified S-(O-A or A-O). Most existing expert systems do not have the capability of changing the current state. Most expert systems, like SEVERE, only add new deductions, without being able to backtrack (cancel previously deduced results), or to solve occurring inconsistencies, actions which can be accomplished by an S-(O-A or A-O) type inference engine. An inconsistency typically occurs when two incompatible elements exist at the same time. Both problems are related since backtracking is often performed when an inconsistency is met so that the action of backtracking cancels previous results and among them those related to the inconsistency. A system whose blackboard can be altered at undetermined moments, i.e., when it is required by the 'context' or blackboard content, may be complicated to test, and thus may be less robust than a O-A type inference engine. It may be further complicated because it heavily depends upon the domain and the problem at hand. Moreover, the overall tree of shells has some interesting features which enhance the flexibility of any of its nodes and justifes the simple O-A inference engine of the shell.

The first interesting feature of the tree structure is that the meta-level of a node may correct the possible lack of flexibility of its sub-nodes. For example, the meta-level of a node may control the current state of the sub-node it is about to call and thus, by

defining the first state, considerably influence the characteristics of the current state of the sub-node. The second interesting feature is that a node may be activated several times by its manager, a process which can be seen as the result of a backtrack. These features imply that the tree structure of the overall system can be seen as an S-A-O type expert (though its inference engine does not formally exist!).

What has been argued about the O-A type of the inference engine is true to the extent that the definition of the meta-level, 'pruning and sorting' the object-level knowledge-base is kept. However, at present, the meta-level can perform meta-actions (see '5.1.5') and thus, can also alter the blackboard content, at each cycle of the scheduler. We must recognize that SEVERE was not devised in that spirit and that it may really be awkward to use SEVERE with problems where backtracking has to be performed. This is why creating meta-rules which would alter the blackboard are not recommended and why we prefer to present SEVERE with an O-A type inference engine.

5.2 THE EXPLANATION FACILITY

5.2.1 Introduction

Mycin is not only popular for its implementation of a production-based system but also for its explanation facility. Its capability in handling why-type and how-type questions was first highlighted in (Shortliffe, 76), and inspired other explanation facilities, such as Prospector (Duda & al, 78), because it proved to be a valuable and simple approach to providing information and

feedback. However the choice implies the following assumptions:

1. A sufficient explanation can be given by merely providing a record of the reasoning of the system.

2. The user still must have, beforehand, an understanding of the overall shell. For example, the facility will not provide rationale for the computing of certainty factors.

At present, the explanation facility, though designed in a general way, should be only used by the knowledge-engineer because it lacks a "natural-language interface" and provides explanations about a node and not about the overall system.

5.2.2 Presentation Of The Explanation Facility

The shell contains a justifier interface which collects all the information of interest present in the work-space of the instantiated node and stores it in a file. After a session with the overall system, the user may want explanations about a particular (intermediate-) result by running the facility. The user is then asked for the node of interest, and the following files are consequently consulted:

1. the file containing information about the last session,

2. the knowledge-base,

3. the meta-knowledge base,

4. the file containing the blackboard data peculiar to the node.

When the user requests explanations about "how a particular element was proved", the explanation facility scans the file to find all the inferences the expert system performed which concluded the element. They are all displayed, and the measure of belief and of disbelief is updated for each inference. The updating is performed in the chronological order of the deduction process. The process is straightforward in this case since the system merely displays, one after the other, trace records present in the file.

Questions related to object- or meta-rules and why-type questions about elements, imply a scanning of the file, to find the trace records which contain the element or rule-number of interest. The system then gathers the found records and elaborates an answer. For instance, an answer about a meta-rule will contain not only a presentation of the meta-rule, but also the set of all the object-elements the meta-rule may have indirectly deduced by the strategy to which it contributed. Furthermore, an explanation about how these elements were derived is also provided.

The need for a simple but efficient explanation facility leads to the choice of Mycin's explanation facility. Since Mycin does not have a meta-level, a simple adaptation of Mycin's explanation facility for the meta-level was devised. The handling of meta-rules and meta-elements are similar to the handling of object-rules and elements. However, since the meta-level strongly influences the object-level, an explanation dealing with the meta-level provides

information the particular requested element or rule and also information about what was implied at the object-level, as mentioned in the above example.

5.3 PRESENTATION OF THE KNOWLEDGE ACQUISISTION SYSTEM

5.3.1 Introduction

The knowledge acquisition system is based upon Teiresias (Davis & Lenat, 82) (see the presentation of Teiresias and its "models", in '2.3.3'). This system was chosen because of its capability to guide the human expert when entering a new rule, by use of models. At present, only the updating of the object-level knowledge base is implemented. The meta-level contains very specialized and procedural concepts which have to be carefully conceived. It was felt that the entering of code should be done directly. The present system, though inspired from Teiresias, does not automatically create and update models, as Teiresias does. They have to be updated manually. This is a definite drawback for an extensive use, but is sufficient for this study since we are more interested in proving the efficiency and adequacy of the proposed explanation facility. However, despite its limitations, the facility is nonetheless of help, even in the case no model matches the entered rule, because of the guidance it provides and because it checks the existence of the entered elements.

5.3.2 Presentation Of The Knowledge Acquisition System

The program contains three phases:

1. Phase 1: The user is prompted for the name of the node to update. Once entered, the system consults its knowledge base.

2. Phase 2: The user enters a rule and its elements are checked for existence.

   If all the elements are recognized by the system, the system executes phase 3.

   If an element is not recognized by the system, the user is prompted to confirm or invalidate its entry. If confirmed, the user is then prompted to give some information about the element. Once confirmed and defined, the element is memorized by the node. If invalidated, the system begins again at phase 1.

3. Phase 3: The system looks for the model which best fits the rule and informs the user about the content of the model. The system presents elements which are often encountered in other rules of this model, and thus, may be missing in the entered rule. Finally, the rule is stored in the knowledge base.

5.4 CHARACTERISTICS OF THE SHELL

(Aubert, 84) proposed a list of the critical features of knowledge-based systems, and they are now used to characterize SEVERE:

1. Generality of the control structure: At the node level, the possibility of backward and forward chaining exists. The control structure of SEVERE is of type (S-)O-A at the node level, whereas the overall tree structure can be seen as an S-O-A inference engine.

2. Types of knowledge representation: Only one form of representation is available in SEVERE: the production system.

3. Supplesness in the writing and updating of rules: An elementary knowledge acquisition system is provided.

4. Existence of an explanation facility: A simple interactive explanation facility, similar to Mycin's explanation facility is provided for the object-level and the meta-level of any node, but does not include "why not"-type questions at present. The explanation facility deals with only one node at a time.

5. Capacity of the system to run step by step: The Mprolog interpreter does not allow the running in trace mode of a program. However, a facility provided by M-prolog (Gur-Arie, 85) allows, under certain constraints, to trace the step-by-step running of one instantiated shell.

6. Possiblity of changing at any time previously entered data: Since the shell is not intended for interactive session, the inputs cannot be changed.

7. Existence of a natural language interface with a spelling corrector: SEVERE does not contain a natural language inteface; mispelling are indirectly detected as the entered elements cannot be found in the list of known elements.

8. Existence of interfaces with high level language : Interfaces with Fortran, Assembler and the command language of the host computer are provided by M-prolog.

9. Inclusion of variables in rules: The nature of Prolog simplifies the introduction of variables in rules.

10. Existence of file of cases allowing the testing, in batch mode, of a knowledge base (for productivity purpose): Test cases are still to be designed. What is provided is a simulation highlighting the characteristics of the system.

11. Existence of a graphic interface: No graphic interface is provided.

5.5 SUMMARY

SEVERE has been presented and its main characteristics enumerated. Rationales for the main design choices were also provided. However, only extensive experiments can validate SEVERE, and the hierarchically organized approach adopted. The purpose of the following chapter is thus to highlight the capabilities of SEVERE in handling an important domain of remote sensing which is the detection of changes in forested areas.

CHAPTER 6

INSTANTIATION OF THE SHELL :

A KNOWLEDGE-BASED SYSTEM FOR

THE DETECTION OF CHANGES

IN REMOTE SENSING

6.1 PRESENTATION OF THE PROBLEM AND OVERVIEW OF THE PROPOSED SOLUTION

Two instantiated experts with a limited knowledge are presented. The instantiation highlights how two nodes interact and in more detail the functioning of one of the node. The chosen problem involves detecting among a set of five forested regions the ones which may have undergone a change. Given a set of supposedly homogeneous regions belonging to the same class at a given date T1, the goal of the system is then to detect the regions which no longer belong to their original class by using a Landsat image taken at a more recent date T2. The proposed example is a simulation of a problem typical for remote-sensing. The low-level processes called by the leaf-nodes and presented in chapter four are LDIAS programs. LDIAS stands for "Landsat Digital Image Analysis System" and designates a set of computers and a complex software dedicated to the handling of remote sensing tasks (Fung & Goodenough, 83).

Suppose we are given five regions which do not change between two Landsat pictures (Table 7.1).

Region Number	Class at time T1	Class at time T2	Similar Regions according to SEPAR time T1	time T2
4	class1	class1	[5,6,8,9]	[5,6,8]
5	class1	class1	[4,6,8,9]	[4,6,8]
6	class1	class1	[4,5,8,9]	[4,5,8]
8	class1	class1	[4,5,6,9]	[4,5,6]
9	class1	class1	[4,5,6,8]	none

TABLE 7.1
labels representing a region, and their actual class at the two different times Landsat pictures were taken.

Further, we suppose that the five regions, though quite homogeneous in the image at T1 differ in some respect in the second image taken at T2. For example, one region may be shadowed, or have more moisture.

To get an idea of what happened, a LDIAS program called SEPAR, is used. This program proposes a partition of the set of regions, according to their "similarity", by use of the actual intensity measures of the region pixels available in the the Landsat image taken at time T2. The "similarity" is based on different matrices of measures (a diffusion matrix, a divergence matrix, etc) and on the use of a threshold on these measures. We suppose that SEPAR finds a non-trivial partition of the set of regions using the second Landsat image, namely: [[4,5,6,8] and [9]]. (Table 7.1) In other words, Separ deduced that there exists two sets of regions ([4,5,6,8], and [9]) which are sufficiently dissimilar to prevent their concatenation. The partition does not imply that the regions are truly different, but that is a possibility. For instance, for such a situation, it seems reasonable to suppose that the first set of regions [4,5,6,8]

represents those which did not change and thus, the other set [9] is the one which did change. The expert system handling the change detection process could then try to prove that nothing changed within the first set of regions and that something happened for the second set. This could be attempted by sending the sets of regions to sub-experts specialized in detecting changes or the absence of change. Once this advice is received, the change detection expert could, for example, decide between the two following possibilities which could be embodied as rules:

1. If the expert specialized in no-change is highly confident in the no-change hypothesis, the change detection expert may decide that the evidence is sufficient in this case.

2. If the expert specialized in detecting the existence of a change is only mildly confident in the hypothesis that the second set contains a region which changed, the change detection expert may decide to request the advice of the no-change expert.

If other results are obtained, other decisions may easily be imagined, but are not needed in the present example:

The data were chosen so that the first set gave a good confidence for the no-change hypothesis, and such that the confidence was mild for the change hypothesis for the second set. The change detection expert thus tried to prove the no-change hypothesis for the second set, thus overriding the conclusion given by SEPAR that the second set is of a different nature from the first set, and found good results.

Calling several experts for a same set implies that contradictory results may be found about regions. It is thus necessary to call the arbitrator for the selection of the likeliest result for each region. In the present case, no other task has to be performed, so the session ends.

6.2 HIERARCHICAL ORGANIZATION OF EXPERT SYSTEMS

The proposed problem required the instantiation of a node, called Change-expert, which handles the change-detection process, and three sub-nodes. The first, called Separ-expert, handles results of the SEPAR algorithm; the second, called Ratio-expert, is specialized in detecting the existence of changes; and the third, called Nochange-expert, is devoted to detecting the absence of changes. The name "Ratio-expert" is chosen from an algorithm which detects changes based upon a "ratioing" technique (cf. chapter three). The overall hierarchy is summarized in Figure 6-1 :

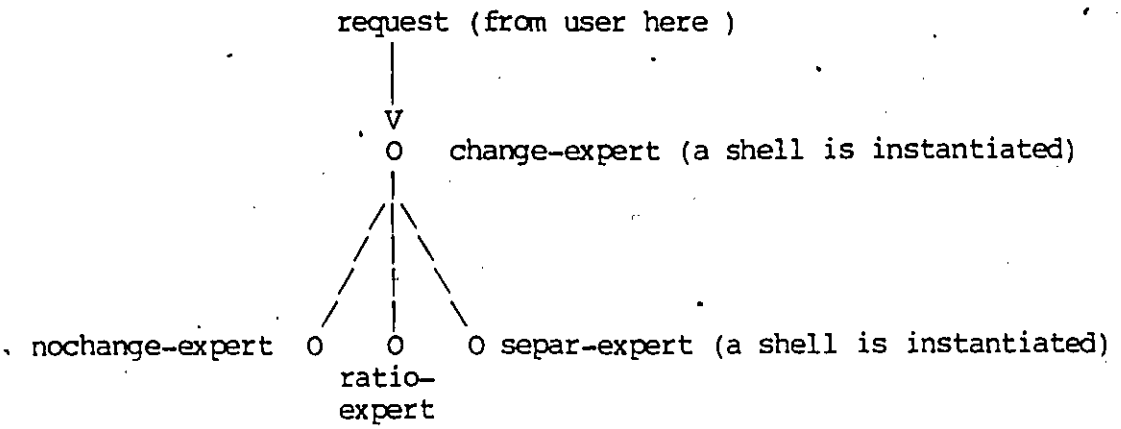

Figure 6-1
Schema of the system simulating
the detection of changes in remote-sensing

Two nodes are actually instantiated: Change-expert, and Separ-expert. Change-expert takes the final decision based upon the information provided by its sub-nodes. The decision is a labeling of the regions of interest, a label being either "change" or "no-change".

To get more information on which to base a decision, "change-expert" calls the sub-nodes "nochange-expert" and "ratio-expert" which in fact are replaced by files which are read and their contents transferred into the "change-expert" blackboard.

6.3 PRESENTATION OF THE INSTANTIATED SHELLS

Change-expert contains seven meta-rules which calls different sub-nodes and modules. It also contains two object rules which draw conclusions about the existence of a clear-cut (a change) or no change. Several interpretations of the occurring changes are possible. To simplify the instantiation, the system has only rules to

conclude "no-change" or "clear-cut". The "clear-cut" hypothesis will represent here the existence of a change in general, though the change-detection expert knows that other hypotheses exist: the arbitrator will note that nothing was deduced about these other hypotheses and will reject them while choosing the best solution.

The second expert called "separ-expert" contains only one meta-rule and one object-rule. In the example, the meta-rule and the object-rule are only created to simulate the process of finding a partition of a set of regions by calling the SEPAR algorithm, and to assess the validity of the proposed partition. In the example, the two rules are merely used to find a partition already present in the blackboard and to send it to "change-expert".

6.4 RULES USED IN THE EXAMPLES

6.4.1 The Meta-rules Of The Change Detection Expert And Their Use

The seven meta-rules of change-expert participated in the design of six strategies, so a strategy was created by concatenating the conclusions of two meta-rules. Table 6-3 presents the rules in the order the meta-level interpreter will choose them.

An example of a coded rule is

```
mknB([activ(separ-expert,reset,X),              /* conclusions */
      put-in-bb(data-base,[regions-of-study(X)])],
     [already-called-lower-nodes(empty),        /* premise     */
      command-upper([detect-ch(exhaustif,X)])],
     1,4).                          /* priority, rule number */
```

This rule means :

if no sub-node has been called yet,

and if the manager's request (the user's request) is to detect all the changes which occurred for the set of regions X,

then ask the lower node "separ-expert" for a new problem concerning the set X

and also put in the part of the blackboard where data are stored the fact "regions-of-study(X)" (this will be helpful for the arbitrator).

The rule number is 4 and its priority is 1. In fact this rule is the first one to be fired and will create the first strategy.

From now on we only present the meaning of the meta-rules. Details are provided in the appendix.

premise elements -----	---> conclusion elements	Rule Number	Priority
1) no sub-node has been called yet 2) manager's request is to detect any change for the regions X	1) ask 'separ-expert' to study the set of regions X 2) put in blackboard the fact 'regions-of-study(X)'	4	1
1) Y is a 'big set of homogeneous regions'	1) ask 'nochange-expert' to study Y	3	2
1) Z is a 'isolated region'	1) ask 'ratio-expert' to study Z	5	2
1) Y was studied by by 'ratio-expert' and it gave mild results	1) ask the 'nochange-expert' to study Y	6	3
1) manager's request is to detect any change for the regions X	1) put all the object-rules available to the interpreter 2) let the interpreter find conclusions for regions X in forward chaining	2	18
1) (nothing)	1) call the arbitrator	1	19
1) (nothing)	1) send message to manager	7	20

TABLE 6.3
PRESENTATION OF THE SEVEN META-RULES
THEIR MEANING, PRIORITY, AND RULE-NUMBER
used in "change-expert". When fired,
a rule is inhibitted and its
conclusions are part of the strategy
currently created

The elements 'big-set-of-homogeneous-regions' and 'isolated-regions' conceptually represent a procedure which would use data from the node. Here, [4,5,6,8] will be assumed to be a 'big-set-of-homogeneous-regions', and [[9]] the set of 'isolated-regions'.

The rules were presented in the order they are stored in the knowledge-base. Such a ordering is the result of different sessions of knowledge acquisition and typically represents what would be found in an actual knowledge-base. For clarity, this order also corresponds to the order of firing of the meta-rules.

The consultation used all the presented rules. and created the six strategies:

[4], [3,5], [6], [2], [1], [7].

Each set contains the meta-rule(s) which created the represented strategy.

6.4.2 The Object Rules In The Change Detection Expert And Their Use

Two rules were created which were:

1. If the current region of study is X
   and the nochange expert found no change occurred in region X
   and it found the result with a confidence greater than .75

   then conclude that no change happened in region X
   with a certainty factor of rule of 1

2. If the current region of study is X

and the ratio-expert found a change occurred in region X

then conclude that a clear-cut occurred in region X

with a certainty factor of rule of 1

The two rules represent the minimum required to conclude whether a change occurred or not, and possibly to create contradictions.

As an example, here is the coded form of the second rule :

```
pot-knB([cc-h(X)],                          /* conclusion */
        [current-region(X),                 /* premise    */
         ratio([true,X])],
        100,7)              /* certainty factor, rule number */
```

The predicate "current-region(X)" is used to instantiate the rule for a given region. The interpreter instantiates the predicate before a scanning of the knowledge base.

6.4.3 The Rules Inside SEPAR Expert

Only one meta-rule, and one object rule were created. Their goal is to read an external file which contains proposals for partitions (such results could be obtained by activation of the conventional program called SEPAR), and takes the best partition for transfer to the change detection expert. Although the separ-expert node at present is very artificial, it was instantiated to demonstrate the interactions between two nodes.

6.4.4 Expertise Embodied In Rules

The process of creating a hierarchy, of defining the concepts or the symbolism of the data is the knowledge engineer's duty, with the help of an expert of the filed. The rules and their meaning, at the object level are typically provided by the expert. The rules presented in '6.4.2' thus represent some knowledge directly given by an expert. The meta-rules are basically given by the field expert but some may be designed by the knowledge engineer. For instance, the meta-rules number 1 and 7 (cf. table 6.3), could be provided by the knowledge engineer whereas the remaining rules would be provided by the expert. We recall that the presented rules and the overall organization as well represent a simulation and are not an actual proposal for handling the detection of changes.

6.4.5 Other Experts

Other experts are called, the no-change expert, the ratio-expert, each of these are represented by files which are read and their content transferred to the change detection expert blackboard.

6.5 STATEMENT OF THE PROBLEM AND ITS SOLUTION

The change detection expert will call in turn the other experts mentioned and for each region, the following facts will be available:

Region Number	Results from no-change expert (tr/f)	Measure Bel	Dis	Results from ratio-expert (true/false)	Measure Bel	Dis
4	TRUE	.95	.05	(NO DATA)	-	-
5	TRUE	.90	.05	(NO DATA)	-	-
6	TRUE	.90	.10	(NO DATA)	-	-
8	TRUE	.90	.20	(NO DATA)	-	-
9	TRUE	.95	.05	TRUE	.35	.10

TABLE 6.4
Results obtained region by region from the no-change expert and ratio-expert

It is expected that regions 4,5,6 will be considered as not having undergone a change, but with varying degrees of confidence. As for region 8, nothing will be deduced: both rules will fail, the first one because of the threshold, the second by lack of data. It is also expected that the conflict for region 9 will be solved in favor of a no-change.

The line of reasoning of the system for region 9 is as follows:

1. SEPAR isolated region 9 from the others. Something could have happened in this set so the ratio-expert is called.

2. The answer provided by ratio-expert is examined. If the occurrence of a change is confirmed but only with a mild degree of confidence, the no-change expert will be requested to study the region. This happens in the present case, because a meta-procedure found that the result from ratio-expert with a certainty factor of:

CF = 0.35 - 0.10 = 0.25

is "mild".

3. The conclusion with highest certainty factor is selected. Since the occurence of no-change is more likely, the expert passes over the partition from SEPAR and considers region 9 similar to the other ones.

6.6 USE OF THE EXPLANATION FACILITY AND OF THE KNOWLEDGE ACQUISITION SYSTEM

The explanation facility and the knowledge acquisition system have been adapted from the Teiresias model (cf. chapter two). Two examples highlighting their capabilities are presented in appendixes B and C. Appendix B presents a session between a user and the explanation facility. Various requests, about the meta-level and the object-level are given. Appendix C presents a session between a knowledge engineer and the knowledge acquisition system. A new rule is entered and upgraded by comparison with two predefined models. We mentioned earlier (chapter two), that no user-interface was provided as such. The explanation facility and the knowledge acquisition system thus interact with the user in predefined and constrained sessions.

CHAPTER 7

CONCLUSIONS

and SUGGESTIONS for FURTHER RESEARCH

The study presented SEVERE, an expert system aimed at creating a hierarchy of expert systems for remote sensing applications. A tree-structured organization was proposed which was instantiated by a two-node system simulating the process of detecting changes in forested areas. The simulation showed how nodes communicate between each other and how a final solution is reached. The study emphasized the importance of the meta-level of a node. The chosen organization implies that a node makes sophisticated strategic decisions influencing the behavior of its son-nodes and through them its descendants at large. The meta-level also contributes to the flexibility of SEVERE by allowing the control structure to make decisions on the choice of the next object to process and then on the nature of the action to perform. The production system was selected to represent knowledge because of the ease in managing and updating the knowledge base it provides. A blackboard is the core of SEVERE's organisation. Though fairly simple, it allows the storing of object-level as well as meta-level data and results. The control structure of the interpreter allows the chaining of rules in forward and in backward mode. A rather simple interface is added to the shell for communication requirements. The interface with outside systems such as geographic data-bases is only outlined. Its implementation in realistic environments will imply the specification of many technical characteristics dependent upon the specific data-bases to link. Finally, a knowledge acquisition system and an explanation facility are also provided.

The approach of designing an expert system shell showed also its constraints. The meta-level is highly procedural, and some tailoring may be needed to optimize SEVERE to the targeted problem. This implies that an instantiation will require a knowledge engineer having a thorough understanding of both SEVERE and of the implemented organization of experts. The procedural aspects facilitates instantiation by eliminating format constraints as well as control constraints but procedural knowledge is hard to update, to validate, and explain. The knowledge acquisition system and the explanation facility were also added to provide tools for instantiations but were not the main subjects of the study. Furthermore, a genuine user interface has still to be designed.

This shows that many enhancements are possible and desirable. It is also likely that realistic instantiations will provide guidelines for more detailed and constrained specifications of the different modules, specifications which would be kept valid for a number of instantiations of SEVERE. Future instantiations will also justify (or invalidate) some of the assumptions made along this study. Among the main issues to be answered, in the context of remote sensing, we can list:

i) The sufficiency of production systems as the sole mean of representing knowledge.

ii) The validity of the hierarchal approach, from different perpectives: the cost of the overhead should be assessed; expert systems with overlapping domains of competence should be efficiently handled by a manager-node; different experts should cooperate in the building of a final solution; the capability of interacting with

outside systems should be estimated; the overall system should be able to create strategies in case of partial failure though no explicit overall control system exists.

iii) The flexibility and power of each instantiated shell in handling the various tasks of remote sensing; in particular, the appropriateness of the fact that the shell is retricted to classification problems and the flexibility of the control structure should be assessed.

Finally, we point out the importance of designing a genuine user-interface which will provide critical support in the instantiation process of SEVERE.

CHAPTER 8

REFERENCES

AUBERT, J.P., (1984), "Comparaison de Quelques Moteurs d'Inference". Bulletin de Liaison de la Recherche en Informatique et Automatique. no-97, pp-34-37.

BARR, A. & FEIGENBAUM, E.A., (1981), The Handbook of Artificial Intelligence, (3 volumes), Heuristech Press, Stanford California.

BENNETT, J.S., CREARY, L.A., ENGELMORE, R.M., MELOSH, R.E., (1978), SACON: A Knowledge Based Consultant in Structural Analysis. Heuristic Programming Project Rep. No. HPP-78-23, Computr Science Department, Stanford University.

BRAMER, M.A., (1982), "A Survey and Critical Review of Expert Systems Research", in Introductory Readings in Expert Systems, Michie D., Editor, Gordon and Breach Science Publishers, pp.3-29.

BRAUN, F., (1984), Techniques in Automatic Processing of Landsat Imagery for Forestry Applications. University of Ottawa, Electrical Engineering Dept, internal memo.

BRAUN, F. & GOLDBERG M., (1985), Un Systeme General pour la Construction d'une Hierarchie d'Experts en Teledetection, University of Ottawa, Electrical Engeneering Department, internal memo.

BRAYER, J.M., SWAIN P.H., FU K.S. (1977). "Modeling of Earth Resources Satellite Data". In Syntactic Pattern Recognition Applications, K. S. FU editor, Springer-Verlag Berlin, 1977, pp.215-242.

BUCHANAN, B.G., & al.,(1969), "Heuristic Dendral: A Program for Generating Explanatory Hypotheses in Organic Chemistry". In B. Metzler and D. Mitchie (Eds.), Machine Intelligence 5. Edinburgh: Edinburgh University Press, pp.253-280.

BUCHANAN, B.G., (1982), "New Research on Expert Systems". In Machine Intelligence 10, Hayes J.E., Michie D., Pao Y-H, Editors, pp.269-299, Halsted Press: a division of John Wiley & Sons.

BUCHANAN, B.G., SHORTLIFFE E.H. (1984), Rule-Based Expert Systems. The Mycin Experiments of the Stanford Heuristic Programming Project, Addison-Wesley Publishing Compagny.

BURNS, Gregory S. & JOYCE, Armond T. (1981), "Evaluation of Land Cover Change Detection Techniques Using Landsat MSS Data", Pecora 7 Symposium, Sioux-Falls, South Dakota, pp.252-260.

BURSTALL, R.M., COLLINS, A., POPPLESTONE, R.J., (1971), Programming in Pop-2, Edimburgh University Press, Edimburgh.

CLOCKSIN, W.F. & MELLISH, C.S., (1981), Programming in Prolog. Springer-Verlag Berlin Germany.

COLMERAUER, A., (1984), "Prolog, Langage de l'Intelligence Artificielle", La Recherche, No. 158, pp.1104-1114, Sept. 1984.

COLWELL, John E., WEBER F.P., (1981), "Forest Change Detection", 15th Symposium on Remote Sensing of Environment, pp.839-852.

CORDIER, M.O., (1984), "Les Systemes Experts", La Recherche, No. 151, pp. 60-70, Janvier 84.

DAVIS, R. & LENAT, D.B., (1982), Knowledge-Based Systems in Artificial Intelligence, Mac Graw Hill Inc.

DUDA, R.O., GASCHNIG, J., HART, P.E., KONOLIGE, K., REBOH, R., BARRETT, P., SLOCUM, J., (1978), Development of the PROSPECTOR consultation system for mineral exploration, Final report, SRI projects 5821 and 6415, SRI-International, Inc., Menlo park, Calif.

DURFEE, E.H., LESSER, V.R., CORKILL, D.D., (1985), "Coherent Cooperation Among Communicating Problem Solvers", to be published.

EGHBALI, Hassan J., (1979), "K-S Test for Detecting Changes from Landsat Imagery Data". IEEE Transactions on S.M.C., vol-SMC 9, No. 1, pp.1-79.

ENGVALL, John E., TUBBS, Jack D., HOLMES, Quentin A., (1977), "Pattern Recognition of Landsat Data Based upon Temporal Trend Analysis", Remote Sensing of Environment 6 , Elsevier Noth-Holland Inc., pp.303-314.

ERMAN, L.D., HAYES-ROTH, F., LESSER, V.R., REDDY, D.R., (1980), "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", Computing Surveys, vol.-12, No. 2, pp.213-253.

FORGY, C.L., (1979), On the Efficient Implementation of Production Systems, Ph. D., Dept. Computer Science. Carnegie- Mellon Univ., Pennsylvania, Feb. 1979.

FORGY, C., McDERMOTT, J., (1977), "OPS, a Domain-Independant

Production System Language", IJCAI 5, pp.933-939.

FU, K.S., (1975), "Pattern Recognition in Remote Sensing of the Earth's Resources", IEEE Transactions on Geosc. Elec., vol.-14, No. 1, pp.10-18.

FUNG, K.B., GOODENOUGH, D.G., (1983), "LDIAS System Integration and Processing Time", Proc. Cray Research Image Processing Conference, Minneapolis, July, 1983.

GEVARTER, W.B., (1983), "Expert Systems: Limited but Powerful", IEEE Spectrum, August 83, pp.39-45.

GOLDBERG, M., GOODENOUGH, D., ALVO, M., KARAM, G., (1985), "A Hierarchical Expert System For Updating Forestry Maps with Landsat Data", Proceedings of the IEEE, june 85.

GONZALEZ, M. & WINTZ, M., (1977), Digital Image Processing, Addison-Wesley Publishing Company, Reading, Massachussettes

GROJNOWSKI, M., (1981), Elements de Probabilite, Note de Cours de l'Ecole Nationale Superieure des Telecommunications de Paris.

GRUMBACH, A., (1982), Introduction a l'Intelligence Artificielle, Note de Cours de l'Ecole Nationale Superieure des Telecommunications de Paris.

GUR-ARIE, Y, (1984), Introduction to Programming with M-Prolog, Lecture Notes, Logicware Inc., Toronto.

GURNEY, Charlotte M., (1980), "Threshold Selection for Line Detection Algorithms", IEEE transactions on Geosc. Elect. vol. 18, No. 4, pp.204-211.

GURNEY, Charlotte M., (1982), "The Use of Contextual Information to Detect Cumulus Clouds and Cloud Shadows in Landsat Data", International Journal of Remote Sensing, vol-3,n-1, pp51-62.

GURNEY, Charlotte M., (1983), "The Use of Contextual Information in the Classification of Remotely Sensed Data", Photogrametric Engineering and Remote Sensing, vol. 49, No. 1, pp.55-64.

HAYES-ROTH, F. & al., (1983), Building Expert Systems, Addison-Wesley Publishing Company, Inc.

HEGYI, F., QUENET, R.V. (1982), "Updating the Forest Inventory Data Base in British Columbia", Remote Sensing for Resource Management, C.J. Johannsen and J.L. Sanders, Editors, SCSA 1982.

HORD, R.M., (1982), Digital Image Processing of Remotely Sensed Data, W. Rheinboldt, Editor, Academic Press Inc., London.

HUSSON, J.M., (1985), "Pilotex: Un Systeme Revolutionnaire de Conduite de Process", Micro & Robots, No.16, Mars 85, pp.56-60.

JAYNES, J., (1976), The Origin of Conciousness in the Breakdown of the Bicameral Mind, Houghton Mifflin, Boston.

JOYCE, Armond T., IVEY, John H., BURNS, Gregory S.. (1980), "The Use of Landsat MSS Data for Detecting Land Use Changes in Forestland", 14th Symposium on Remote Sensing of Environment, San Jose, Costa-Rica, pp.979-988.

KAUTH, R.J., THOMAS, G.S. (1976), "The Tasseled Cap -- A Graphic Description of the Spectral-Temporal Development of Agricultural Crops as Seen by Landsat", Machine Processing of Remote Sensing Data Symposium, W.Lafayette, Indiana, pp.4B-41--4B-51.

KAY, A., GOLDBERG, A., (1977), "Personal Dynamic Media", Computer, No. 10, pp.31-41.

KETTIG, R.L.,LANDGREBE, D.A., (1976), "Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects", IEEE transactions on Geoscience Electronics, vol. GE-14, No.-1, 19-26.

KUNZ, J. & al., (1978), A Physiological Rule-Based System for Interpreting Pulmonary Function Test Results, Heuristic Programming Project Rep. No. HPP-78-19, Computer Science Dept., Stanford University.

LAURENT, J.P., (1984), "La Structure de Controle des Systemes Experts", Techniques et Sciences de l'Informatique, vol-3, no-3, pp.161-177.

LAURIERE, J.L., (1984), "Un Moteur d'Inference pour Systems Experts en Logique du Premier Ordre: Snark/ Symbolic Normalized Acquisition and Representation of Knowledge", Bulletin de la Liaison de la Recherche en Informatique et Automatique, No. 97/1984, pp. 24-28.

LAURIERE, J.L., LAGRANGE, M.S., RENAUD, M., (1982), Simulation d'un Raisonnement Archeologique, fasc. 1: Description du Systeme Snark; Lab. d'Informatique pour les Sciences de l'Homme, CNRS, Paris, juillet 1982.

LEA, W. & SHOUP, J., (1980), "Specific Contributions of the ARPA SUR Project", Lea W., Editor, in Trends in Speech Recognition, Englewood Cliffs, N.J., Prentice-Hall, pp.3-18.

LIKENS, W., MAW, K., SINNOTT, D., (1982), Final Report : Landsat Land Cover Analysis Completed for CIRSS/San Bernardino County Project, Nasa Technical Memorandum 84244.

LINTZ, J, SIMONETT, D.S., (1976), Remote Sensing of Environment, Addison-Wesley Publishing Company, Inc.

LOGAN, T.L., STRAHLER, A.H. (1982) . "Optimal Landsat Transforms for Forest Applications", 16th Symposium on Remote Sensing of Environment, Buenos Aires, Argentina, pp.455-468.

MALILA, William, A. (1980), "Change Vector Analysis: An Approach for Detecting Forest Changes with Landsat", Machine Processing of Remote Sensing Data Symposium, pp.326-335.

METZLER, M. D., CICONE, R. C., JOHNSON, K. I. (1983), " Experiments with an Expert-Based Crop Area Estimation Technique for Corn and Soybean", 17th Symposium on Remote Sensing of Environment, Ann-Arbor, Michigan, pp.965-972.

METZLER, M. D., CICONE, R. C., JOHNSON, K. I. (1982), " The Evaluation of a Semi-Automated Procedure for Classifying Corn and Soybeans without Ground Data", Machine Processing of Remote Sensing Data Symposium, W.Lafayette, Indiana, pp.108-115.

MISRA, P.N., WHEELER, S.G. (1978), "Crop Classification with Landsat Multispectral Scanner Data", Pattern Recognition, vol. 10, pp. 1-13.

MOONEYHAN, D.W. (1983), "The Potential of Expert Systems for Remote Sensing Applications", International Geoscience and Remote Sensing Symposium, (IGARSS '1983), San Francisco, pp.4.1-4.5.

MULSANT, B., SERVAN-SCHREIBER, D., (1984), "Knowledge Engineering: A daily Activity on Hospital Ward", Computers and Biomedical Research 17, pp.71-91.

NAGAO, M., MATSUYAMA, T., (1981), A Structural Analysis of Complex Aerial Photographs, Plenum Press, New York.

NEWELL, A., SHAW, J.C., SIMON, H.A., (1957), "Programming the Logic Theory Machine", Proceedings of the Western Joint Computer Conference, pp.230-240.

OPPACHER, M., (1984), "Course Notes about Expert Systems", Carleton University, department of Computer Sciences, fall 1984.

PARCY, J.P., (1982), Un Systeme Expert en Diagnostic sur Reacteurs a Neutrons Rapides. These de 3eme cycle. Aix-Marseilles II.

PARK, A.B., HOUGHTON, R.A., HICKS, G.M. (1983), "Multitemporal Change Detection Techniques for the Identification and Monitoring of Forest Disturbances", 17th Symposium on Remote Sensing of Environment, Ann Arbor, Michigan, pp.77-97.

PARK, A.B., WESCOTT, T.F., ROYAL, J.A., (1982), "Urban Encroachment on Agricultural Land", International Symposium on Remote Sensing of Environment 1st Thematic Conference: Remote Sensing of Arid and Semi-Arid Lands, Cairo, Egypt, pp.217-226.

PAVLIDIS, T., (1979), "Hierarchies in Structural Pattern Recognition", Proceedings of the IEEE, Vol.-67, No.-5, pp.737-744.

PEREIRA, F.C.N. & WARREN, D.H.D., (1980), "Definite Clause Grammars for Language Analysis- A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, No. 13, pp.231-278, North-Holland Publishing Company.

PESTRE, C. R., MALILA, W. A., (1983), "Using Knowledge of Agricultural Practices to Enhance Through-the-Season Interpretation of Landsat Data", 17th Symposium on Remote Sensing of Environment, Ann Arbor, Michigan, pp.957-962.

PITRAT, J., (1984), "La Genese des Systemes Experts", Bulletin de la Liaison de la Recherche en Informatique et Automatique., No.-97, pp.1-8.

POPESCU, R., (1984). Chelem un S.E. pour Trouver la Ligne de Jeu du Declarant au Bridge. These de 3eme cycle. Institut de Programmation.

POPLE, H., (1977), "The Formation of Composite Hypotheses in

diagnosis Problem Solving — An Exercise in Synthetic Reasoning". IJCAI 5, pp.1030-1037.

REDDY, R., ERMAN, L., FENNELL, R., NEELY, R., (1976), "The Hearsay Speech Understanding System : An Example of the Recognition Process". IEEE Transactions on Computers. Vol.-25, pp.427-431.

ROUSSEL, P., (1975), Prolog—Manuel de Reference et d'Utilisation, Groupe d'Intelligence Artificielle, UER de Luminy, Universite d'Aix-Marseilles II.

RYCHENER, M.D., ALCANTARA, R.B., SUBRAHMANIAN, E., (1984), "A Rule-Based Blackboard Kernel System: Some Principles in Design, Proccedings of the IEEE Workshop on Principles of Knowledge-Based Systems. pp.59-64.

SHAFER, G., (1976), A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ.

SHIMODA, H.,FUKUE, K., KINOSADA, Y., SAKATA, T. (1983), "Landuse Mapping and Change Detection with the Aid of Syntactic Approach", 17th Symposium on Remote Sensing of Environment, Ann Arbor, Michigan, pp.973-979.

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

SMITH, D.H. & CARHART, R.E., (1978), "Structure Elucidation Based on Computer Analysis of High and Low Resolution Mass Spectral Data", M.L. Gross, Editor, in High Performance Mass Spectroscopy: Chemical Applications, American Chemical Society, Washington D.C., 325.

SWAIN, P.H., DAVIS, S.M., (Editors) (1978), Remote Sensing, the Quantitative Approach, Mc Graw-Hill.

SWAIN, P.H., WARDEMAN, S.B., TILTON, J.C., (1981), "Contextual Classification of Multispectral Image Data", Pattern Recognition, vol-13, n-6, pp.398-429.

TINNEY, L.R., SAILER, C., ESTES, J.E. (1983), "Applications of Artificial Intelligence to Remote Sensing", 17th Symposium on Remote Sensing of Environment, Ann Arbor, Michigan, pp.255-269.

TODD, William J. (1977), " Urban and Regional Land Use Change Detected by Using Landsat Data", Journal Research U.S. Geological

Survey, Vol. 5, No. 5, pp.529-534.

WHEELER, S.G.,MISRA, P.N. (1980), "Crop Classification with Landsat Multispectral Scanner Data II", Pattern Recognition, Vol. 12, pp. 219-228.

WILKINS, D.E., (1982), "Using Knowledge to Control Tree-Searching", Artificial Intelligence Journal, No. 18, pp.1-51.

WILLIAMS, D.L., STAUFFER, M.L., LEUNG, K.C.,(1979), "A Forester's Look at the Application of Image Manipulation Techniques to Multitemporal Landsat Data", Machine Processing of Remote Sensing Data Symposium, pp.368-375.

WINSTON, P.H., (1977), Artificial Intelligence, Addison-Wesley, Inc., Reading, Mass.

WINSTON, P.H. & PRENDERGAST, K.A. (Editors), (1984), The AI Business Commercial Uses of Artificial Intelligence, The MIT Press, Cambridge Ma., London England.

WOODCOCK, C.E., FRANKLIN, J., STRAHLER, A.H., LOGAN, T.L. (1982), "Improvements in Forest Classification and Inventory Using Remotely Sensed Data", 16th Symposium on Remote Sensing of Environment, Buenos Aires, Argentina, pp.963-974.

ZADEH, L.A., (1979), "Approximate Reasoning based on fuzzy logic", International Joint Conference on Artificial Intelligence 6, pp.1004-1010.

APPENDIX A

THE SHELL

A.1 PRESENTATION OF A SESSION FOR DETECTING CHANGES IN REMOTE SENSING

A.1.1 Notations

A.1.1.1 Blackboard Notation

The blackboard is represented by a set of statements beginning with "bb" and with the first argument among the list :

1. agenda-upper (contains the request from the user or the manager).

2. agenda-strat (contains the strategies created by the meta-level).

3. data-base (contains the information available before the running of the first cycle within the scheduler).

4. final-results (contains the final solution to the request).

Each time the meta-level is consulted, two statements are created and added to the top of the blackboard definition :

1. results-by-strategy : contains the elements which are known as being goals of the node and which are deduced in the context of the current strategy. (one statement is created per strategy).

2. intermediary-results : contains all the remaining deduced elements (which are not goals) and the elements deduced by sub-nodes called within a strategy.

To keep a simple and identical way of manipulating the statements of the blackboard, the second argument is a list of elements which always has the same format : a list of two sub-lists. In the case of statements with first argument belonging to the set [data-base, final-results],

the first sub-list is empty and is not used. For the other statements, the first sub-list describes the strategy in the context of which the elements in the second sub-list were found.

A.1.1.2 Knowledge-base Notation

The predicate pot-knB is used for the rules which are not available to the object-level interpreter, and knB for the rules which are available. (see '5.1.4')

A.1.1.3 Meta-knowledge Base Notation

The predicate is mknB with four arguments.

A.1.1.4 Upper-command Notation

The user's request is :

"detect-ch(exhaustif,[4,5,6,8,9])"

(arbitrarily) meaning :

"detect any change occurred in any region among the set of regions represented by the labels '4,5,6,8,9'". A "reset" flag is added, warning the node the problem is new.

The change-expert's request is "group([4,5,6,8,9])" (arbitrarily) meaning "propose a grouping of the regions labeled '4,5,6,8,9'". A "reset" flag is added, warning the expert the problem is new.

A.1.1.5 Concept Notation

The concepts are the elements manipulated by the meta-level. They are diverse in notation and meaning and will be explained when used.

A.1.1.6 Object Notation

notation : <predicate name>/<number of arguments>

A.1.1.6.1 Change-expert Level

The goals used by change-expert are :

1. fire-h/1 : a fire is detected.

2. cc-h/1 : a clear-cut is detected.

3. regen-h/1 : an regenerating area is detected (a young forested area).

4. noch-h/1 : no change detected.

Each of these goals have one argument containing the region label of study.

The sub-goals used by the expert are :

1. noch/1 : result from the dummy called nochange-expert.

2. ratio/1 : result from the dummy called ratio-expert.

3. separ/1 : result from the instantiated shell separ-expert.

4. regions-of-study/1 : stores the set of regions to study in its argument.

noch/1 and ratio/1 have a list in their argument :

[<true or false>,<studied region label>]

ex. : noch([true,4]) means that, according the nochange-expert, it is true that no-change occurred in region 4.

separ/1 contains a list representing the partition of [4,5,6,8,9].

A.1.1.6.2 Separ-expert Level

The only goal is separ/1 as above.

The only objects are data (supposed results from the SEPAR algorithm) :

separ-low/1

A.2 MYCIN'S THEORY OF BELIEF

Each rule is assigned a certainty factor (a number between -1 and 1), called CF.

The process of updating the measures for a conclusion is as follows :

1. The minimum of all the measures of belief of the elements in the premise is computed, and called Min.

2. The maximum of all the measures of disbelief of the elements in the premise is computed, and called Max.

3. The difference of the previous numbers is computed, and called Diff.

4. "Local" measures of belief called MBloc and of disbelief called MDloc are computed as follows :

** If CF >= 0, then

+ if Diff >= 0, then MBloc = CF * Diff, MDloc = 0

+ if Diff < 0, then MBloc = 0, MDloc = -(CF * Diff)

** If CF < 0 then

+ if Diff >= 0, then MBloc = 0, MDloc = -(CF * Diff)

+ if Diff < 0, then MBloc = CF * Diff, MDloc = 0.

5. If the element is newly deduced then the computed local measures are its measures of belief and disbelief.

If the element was previously proved then its previous measures of belief called MBold and of disbelief called MDold are updated as two new measures called MB and MD as follows :

MB = MBold + MBloc - (MBold * MBloc)

MD = MDold + MDloc - (MDold * MDloc)

A.3 A CONSULTATION SESSION FOR THE DETECTION OF CHANGES IN FORESTED AREAS

A.3.1 Introduction

The present report on a session was created by adding write commands in the modules. Of course, these write commands do not have to exist in actual implementations. It may be useful, for debugging, to keep track of the current actions of the system. But the end-user should not be bothered with this internal information.

Comments were added to explain the writings and are preceeded by a '%'.

A.3.2 Writings Created During A Session

```
**********************************************************

 entering the inter- node manager...

at the beginning, all the nodes are deactivated
 the first node to activate is :change_detection_expert
the first node is activated
leaving the inter-node manager...

**********************************************************

**********************************************************
```

```
entering the change-detection expert node...

looking for manager's request

command is [reset,detect_ch(exhaustif,[4,5,6,8,9])]

preprocessing...

% The node is called for a new problem so it must reset.
% This is performed by a preprocessing

present content of the bb:

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])
bb(agenda_strat,[])
bb(data_base,[[],[]])
bb(final_results,[[],[]])

% The blackboard is empty, apart from the manager's request
% (reset is a local flag which is not of use
% after the preprocessing)

present content of the mknb:

% The meaning of the meta-rules will be explained later

mknb([activ(separ_expert,reset,_563),
      put_in_bb(data_base,[regions_of_study(_563)])],
     [already_called_lower_nodes(empty),
      command_upper([detect_ch(exhaustif,_563)])],
     1,4)

mknb([activ(nochange_expert,reset,_563)],
     [exist_big_sets_of_homogeneous_regions(_563)],2,3)

mknb([activ(ratioing_expert,reset,_563)],
     [exist_isolated_regions(_563)],2,5)

mknb([met([pot_knb,4],knb,[[*],1]),demontre(_563)],
     [command_upper([detect_ch(exhaustif,_563)])],18,2)

mknb([call_arbitrator],[dummy],19,1)

mknb([send_message],[dummy],20,7)

mknb([activ(nochange_expert,reset,[_563])],
     [mild_results_by_ratioing_expert(_563)],3,6)


present content of the (potential) knowledge-base:

pot_knb([noch_h(_564)],
        [current_region(_564),
         noch([true,_564]),
```

```
         greater(cf(noch([true,_564]),_565),75)],
        100,6)

pot_knb([cc_h(_564)],
        [current_region(_564),ratio([true,_564])],
        100,7)

 preprocessing performed

% The scheduler always performs the same cycle :
%     preprocessing
%     consultation of the meta-level
%     executution of the strategy designed by the meta-level

 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current meta-rule is :

[activ(separ_expert,reset,_599),
 put_in_bb(data_base,[regions_of_study(_599)])]

 <-----

[already_called_lower_nodes(empty),
command_upper([detect_ch(exhaustif,_599)])] 1

%This rule means :
%
%if      no sub-node has been called yet
%and if  the manager's request is to detect all the
%        occurred changes  in the set of regions : _599
%
%then
%
% activate the separ-expert and request a grouping
% of the set of regions _599,
% and put in the blackboard, the set of regions of study

 The mrule is verified;
 its conclusions are entered in the  agenda

leaving the meta-level interpreter...

% only one meta-rule of priority 1 is firable
```

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[activ(separ_expert,reset,[4,5,6,8,9]),
 put_in_bb(data_base,[regions_of_study([4,5,6,8,9])])]


**********************************************************

 entering the inter- node manager...

 ...deactivating change_detection_expert and
    activating   separ_expert

leaving the inter-node manager...

**********************************************************

************************************************************

entering the separ-expert node...


 preprocessing...

 present content of the bb:

bb(agenda_upper,[])
bb(agenda_strat,[])
bb(data_base,[[],[]])
bb(final_results,[])


 present content of the mknb:

% only one meta-rule

mknb([promptdata(exhaustif),
      met([pot_knb,4],knb,[[*],1]),
      demontreforw([separ(_992)],_993)],
     [command_upper([demontreforw([separ(_992)],_994)])],
     1,3)


 present content of the (potential) knowledge-base:

% only one object rule

pot_knb([separ(_990)],
        [exist_a_potential_result_in_bb(separ_low,_990),
         enforce_cf(separ_low,_990)],
```

```
100,1)

 looking for manager's request

 command is [reset,group([4,5,6,8,9])]
preprocessing performed

preprocessing for first pass performed
 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

% The meta-rule is fired and the derived strategy is
% the list of the actions of the rule

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted

++++++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[promptdata(exhaustif),
 met([pot_knb,4],knb,[[*],1]),
 demontreforw([separ(_1229)],_1230)]

%This commands are :
%
% read the external file 'dataext.dat' (which will contain
% outputs from the SEPAR algorithm )
%
% find all the object-level rules
% and put them visible for the object-level interpreter
%
% give the priority to the interpreter which should fire
% object-level rules in a forward mode
%

 The current goal to prouve is : [separ(_1229)]

% Let's recall that the shell always has a prior knowledge
% of the set of possible goals. Here, there is only one to
% find : "separ(_1229)", where _1229. is a partition of a
% set of regions
```

```
entering the interpreter...

 The current rule under study is :

[separ(_1354)]

<——

[exist_a_potential_result_in_bb(separ_low,_1354),
 enforce_cf(separ_low,_1354)] 100

%This object-rule means :
%
% if the result is already in the data-base part of the
% blackboard,
% and if ... the remaining enforces the values of the measures
% of belief and disbelief of the conclusion. This is done because
% the  resulting grouping is a data for the manager which is given
% and sure.
%
%then
%
% the goal, to find a grouping is found


exist_a_potential_result_in_bb(separ_low,[[4,5,6,8],[9]])
is verified
enforce_cf(separ_low,[[4,5,6,8],[9]]) is verified

 The rule is verified,
 its conclusions are enterred in bb

[separ([[4,5,6,8],[9]])] is in the data_base,and
 it does not have to be reproved,so evth is proved!

% Since the expert system found the goal, and since
% the system does not have to prove the goal again
% (it is a characteristic of the present goal which
% is known by the system)
% ... then there is no need to go on scanning the
% knowledge base

leaving the interpreter...
```

```
the search for the current goal :

[separ([[4,5,6,8],[9]])]

% by firing the rule, the system instantiated the variables
% in the rule and found the present result

is finished in the chosen strategy
```

```
the commands selected by the meta-level
are completed. Let's find another strategy...


 present content of the bb :

bb(results_by_strategy,[[3],
                        [[separ([[4,5,6,8],[9]]),
                         [1,100,0]]]])

% by strategy defined by the list of fired meta-rules : [3]
% the system found the goal separ([[4,5,6,8],[9]]) by
% firing the object-level rule number 1 with a
% measure of belief of 1 (100) and a measure of
% disbelief of 0

bb(data_base,[[],[[separ_low([[4,5,6,8],[9]]),52,0],
                  [separ_low([[4,5,6],[8],[9]]),48,0]]])

% by consulting the external file, the system  entered the
% content of the file into the blackboard. Two results were
% found. The system will choose later on the best one and
% if separ-expert is later called again, for the same problem
% it will find the second result (since it would be useless
% to send a result alredy sent

bb(intermediary_results,[[3],[]])

% no intermediary result was found

bb(agenda_strat,[[[3],
                 [promptdata(exhaustif),
                  met([pot_knb,4],knb,[[*],1]),
                  demontreforw([separ(_1585)],_1586)]]])

% stores the strategies

bb(final_results,[])

bb(agenda_upper,[demontreforw([separ(_1585)],_1586)])


 present content of the knowl. base :

knb([separ(_1606)],
    [exist_a_potential_result_in_bb(separ_low,_1606),
     enforce_cf(separ_low,_1606)],100,1)


 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...
```

```
 no more relevant strategy to perform...

% the arbitrator is called and the best result sent to the
% manager

There remains to find the best result among :

[[separ([[4,5,6,8],[9]]),[[[3],1,100,0]]]]

 which is :[separ([[4,5,6,8],[9]]),[[[3],1,100,0]]]

The system found that [separ([[4,5,6,8],[9]]),
                       [[[3],1,100,0]]]

was the more likely result

End of the present consultation for the node

 final content of the bb :

bb(final_results,[separ([[4,5,6,8],[9]]),
                  [[[3],1,100,0]]])

bb(results_by_strategy,[[3],
                        [[separ([[4,5,6,8],[9]]),
                         [1,100,0]]]])

bb(data_base,[[],[[separ_low([[4,5,6,8],[9]]),52,0],
                  [separ_low([[4,5,6],[8],[9]]),48,0]]])

bb(intermediary_results,[[3],[]])

bb(agenda_strat,[[[3],
                 [promptdata(exhaustif),
                  met([pot_knb,4],knb,[[*],1]),
                  demontreforw([separ(_1804)],_1805)]]])

bb(agenda_upper,[demontreforw([separ(_1804)],_1805)])


leaving the separ-expert node...

*********************************************************

...deactivating separ_expert and
   activating   change_detection_expert

leaving the inter-node manager...

*********************************************************

% the system resumes its running at the change-detection
% expert level

the commands selected by the meta-level
are completed. Let's find another strategy...
```

```
 present content of the bb :
bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]])

% the regions of study are [4,5,6,8,9]. This fact was
% was put into the blackboard by an action of the
% previously defined strategy

bb(intermediary_results,[[4],
                         [[separ([[4,5,6,8],[9]]),
                           separ_expert,100,0]]])

% During the execution of the (previous) strategy defined
% by the set of fired rules : [4]
% the system found the result separ([[4,5,6,8],[9]])
% by calling the sub-node separ-expert, thus with
% a measure of belief of 1 (100) and of disbelief of
% 0

bb(results_by_strategy,[[4],[]])

bb(agenda_strat,[[[4],
                 [activ(separ_expert,reset,[4,5,6,8,9]),
                  put_in_bb(data_base,
                            [regions_of_study([4,
                                               5,
                                               6,
                                               8,
                                               9])])]]])

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])

bb(final_results,[[],[]])

 present content of the knowl. base :

% Since there has not been any action to make
% available to the interpreter the rules, the "visible"
% part of the knowledge base is still empty


 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current meta-rule is :
[activ(nochange_expert,reset,_1957)]
```

```
<-----

[exist_big_sets_of_homogeneous_regions(_1957)], 2

%
%this rule means :
%
% if in the grouping there is a big set of homogeneous regions,
% (it is assumed that [4,5,6,8] is a big set with
% regions of the  same type (since they were grouped together
% by the Separ algorithm)
%
%then activate the no-change expert for the set
%

 The mrule is verified;
 its conclusions are entered in the  agenda

% the system finds other meta-rules of priority 2. If they
% are firable, their conclusions are added to the set of
% the previously found.

the current meta-rule is :

[activ(ratioing_expert,reset,_1957)]

<-----

[exist_isolated_regions(_1957)] 2

%this rule means :
%
% if there exists an isolated region in the grouping
% ([9] is an example...)
%
% then
%
% call the 'ratioing-expert' for the region
%

 The mrule is verified;
 its conclusions are entered in the  agenda

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted

++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[activ(nochange_expert,reset,[[4,5,6,8]]),
 activ(ratioing_expert,reset,[9])]
```

```
*********************************************************

 entering the inter- node manager...

 ...deactivating change_detection_expert  and
    activating   nochange_expert

dummy call ;
a file is read and the data transferred in the bb

...deactivating nochange_expert  and
   activating   change_detection_expert

leaving the inter-node manager...

*********************************************************

*********************************************************

 entering the inter- node manager...

 ...deactivating change_detection_expert and
    activating   ratioing_expert

dummy call ;
a file is read and the data transferred in the bb

...deactivating ratioing_expert and
   activating   change_detection_expert

leaving the inter-node manager...

*********************************************************

the commands selected by the meta-level
are completed. Let's find another strategy...


 present content of the bb :

bb(intermediary_results,[[3,5],
                         [[noch([true,4]),
                           nochange_expert,90,5],
                          [noch([true,5]),
                           nochange_expert,90,5],
                          [noch([true,6]),
                           nochange_expert,90,10],
                          [noch([true,8]),
                           nochange_expert,90,20],
                          [ratio([true,9]),
                           ratioing_expert,35,10]]])

% These are the results read from the dummy calls

bb(results_by_strategy,[[3,5],[]])
```

```
bb(agenda_strat,[[[4],
                 [activ(separ_expert,reset,[4,5,6,8,9]),
                  put_in_bb(data_base,
                            [regions_of_study([4,
                                               5,
                                               6,
                                               8,
                                               9])])]],
                [[3,5],[activ(nochange_expert,
                              reset,
                              [[4,5,6,8]]),
                        activ(ratioing_expert,
                              reset,[9])]]).

bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]]).

bb(intermediary_results,[[4],[[separ([[4,5,6,8],[9]]),
                               separ_expert,100,0]]]).

bb(results_by_strategy,[[4],[]]).

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]]).

bb(final_results,[[],[]]).
```

present content of the knowl. base :

preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++

consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current [illegible]-rule is :

[activ(nochange_expert,reset,[_2336])]

<----- [

mild_results_by_ratioing_expert(_2336)] 3

```
%this rule means :
%
% if the answer of the ratio-expert for a given region
% is a mild confirmation
% ( for instance ratio([true,9]) is a mild result since the
```

```
% the confidence measure is 35 - 10 = 25 (0.25)
%
% then
%
% call the no-change expert for the same region
%

 The mrule is verified;
 its conclusions are entered in the  agenda

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[activ(nochange_expert,reset,[[9]])]

**********************************************************

 entering the inter- node manager...

 ...deactivating change_detection_expert and
    activating   nochange_expert

dummy call ;
a file is read and the data transferred in the bb

 ...deactivating nochange_expert and
    activating   change_detection_expert

leaving the inter-node manager...

**********************************************************

the commands selected by the meta-level
are completed. Let's find another strategy...


 present content of the bb :

bb(intermediary_results,[[6],[[noch([true,9]),
                               nochange_expert,95,5]]])

% this new result is added to the blackboard by the last
% dummy activation of the ratioing-expert node

bb(results_by_strategy,[[6],[]])

bb(agenda_strat,[[[4],
                 [activ(separ_expert,reset,[4,5,6,8,9]),
                 put_in_bb(data_base,
```

```
                                    [regions_of_study([4,
                                                       5,
                                                       6,
                                                       8,
                                                       9])])]],
                          [[3,5],[activ(nochange_expert,
                                        reset,
                                        [[4,5,6,8]]),
                                  activ(ratioing_expert,
                                        reset,[9])]],
                          [[6],[activ(nochange_expert,reset,[[9]])]]])

bb(intermediary_results,[[3,5],[[noch([true,4]),
                                 nochange_expert,90,5],
                                [noch([true,5]),
                                 nochange_expert,90,5],
                                [noch([true,6]),
                                 nochange_expert,90,10],
                                [noch([true,8]),
                                 nochange_expert,90,20],
                                [ratio([true,9]),
                                 ratioing_expert,35,10]]])

bb(results_by_strategy,[[3,5],[]])

bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]])

bb(intermediary_results,[[4],[[separ([[4,5,6,8],[9]]),
                              separ_expert,100,0]]])

bb(results_by_strategy,[[4],[]])

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])

bb(final_results,[[],[]])

 present content of the knowl. base :


 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current meta-rule is :

[met([pot_knb,4],knb,[[*],1]),demontre(_2683)]
```

```
<-----

[command_upper([detect_ch(exhaustif,_2683)])] 18

% If the command is to detect everything possible for
%.the set of region _2683,
% then
% put available all the object-rules to the
% the object-level interpreter and
% fires any rule in a forward mode
%

 The mrule is verified;
 its conclusions are entered in the agenda

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted

++++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[met([pot_knb,4],knb,[[*],1]),demontre([4,5,6,8,9])]

____________________________________________________

entering the object level interpreter...

 The current goal to prouve is :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 for the region :4

 The current rule under study is :

[noch_h(_3303)]

<---

[current_region(_3303),
 noch([true,_3303]),
 greater(cf(noch([true,_3303]),_3304),75)] 100

%this rule means :
%
% if
%the current region is <region number>
%                    (used to instantiate the rule)
%and if
%the no-change expert confirm that no change occurred in the
%region
%
```

```
%and if
%the certainty factor of the confirmation is greater than .75
%
%then
%deduce that no change occurred for the region
%

current_region(4) is verified
noch([true,4]) is verified
greater(cf(noch([true,4]),85),75) is verified

Its premise [current_region(4),
             noch([true,4]),
             greater(cf(noch([true,4]),85),75)]

is verified; so its conclusions [noch_h(4)]
are entered in the bb

 The current rule under study is :

[cc_h(_4866)]

<---

[current_region(_4866),ratio([true,_4866])] 100

%
%this rule means :
%
%if the current region of study is :_4866
%(used to instantiate the rule)
%
%and if the ratio-expert confirm that a change occured
%for the region
%
%then conclude that a clear-cut occurred for the  region


current_region(4) is verified
ratio([true,4]) cannot be proved,the rule fails

 the system found something but

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

is not entirely found.

% The shell tries to infer all the possible goals (this is
% the manager's request)
% since not all them are found,
% but since something was found,
% possibly enabling the firing of
% new rules,
% another scanning of the knowledge-base
% is performed
```

So another scanning of the knowledge-base is performed...

The current rule under study is :

[noch_h(_5158)]

<---

[current_region(_5158),
noch([true,_5158]),
greater(cf(noch([true,_5158]),_5159),75)] 100

%has been already fired

The current rule under study is :

[cc_h(_5159)]

<---

[current_region(_5159),ratio([true,_5159])] 100

current_region(4) is verified
ratio([true,4]) cannot be proved,the rule fails

%nothing was newly deduced so no other scanning
%will be performed

Everything which could be deduced was deduced.

The goal

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

is not entirely found in the chosen strategy.

the search for the current goal :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :4

is finished in the chosen strategy

The current goal to prouve is :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :5

%exactly the same process will happen for the region 5.

The current rule under study is :

[noch_h(_5498)]

```
<---

[current_region(_5498),
 noch([true,_5498]),
 greater(cf(noch([true,_5498]),_5499),75)] 100

current_region(5) is verified
noch([true,5]) is verified
greater(cf(noch([true,5]),85),75) is verified

Its premise [current_region(5),
             noch([true,5]),
             greater(cf(noch([true,5]),85),75)]

is verified; so its conclusions [noch_h(5)]
are entered in the bb

 The current rule under study is :

[cc_h(_7115)]

<---

[current_region(_7115),ratio([true,_7115])] 100

current_region(5) is verified
ratio([true,5]) cannot be proved,the rule fails

 the system found something but

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

is not entirely found.

 So another scanning of the knowledge-base is performed...

 The current rule under study is :

[noch_h(_7407)]

<---

[current_region(_7407),
 noch([true,_7407]),
 greater(cf(noch([true,_7407]),_7408),75)] 100

 The current rule under study is :

[cc_h(_7408)]

<---

[current_region(_7408),ratio([true,_7408])] 100

current_region(5) is verified
ratio([true,5]) cannot be proved,the rule fails
```

```
 Everything which could be deduced was deduced.

 The goal

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 is not entirely found in the chosen strategy.

 the search for the current goal :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :5

 is finished in the chosen strategy

 The current goal to prouve is :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 for the region :6

%exactly the same process will happen for region 6
%Only the confidence measures will change

 The current rule under study is :

[noch_h(_7747)]

<---

[current_region(_7747),
 noch([true,_7747]),
 greater(cf(noch([true,_7747]),_7748),75)] 100

current_region(6) is verified
noch([true,6]) is verified
greater(cf(noch([true,6]),80),75) is verified

Its premise

[current_region(6),
 noch([true,6]),
 greater(cf(noch([true,6]),80),75)]

is verified; so its conclusions [noch_h(6)]
are entered in the bb

 The current rule under study is :

[cc_h(_9405)]

<---

[current_region(_9405),ratio([true,_9405])] 100
```

```
current_region(6) is verified
ratio([true,6]) cannot be proved,the rule fails

 the system found something but

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

is not entirely found.

 So another scanning of the knowledge-base is performed...

 The current rule under study is :

[noch_h(_9697)]

<---

[current_region(_9697),
 noch([true,_9697]),
 greater(cf(noch([true,_9697]),_9698),75)] 100

 The current rule under study is :

[cc_h(_9698)]

<---

[current_region(_9698),ratio([true,_9698])] 100

current_region(6) is verified
ratio([true,6]) cannot be proved,the rule fails

 Everything which could be deduced was deduced.

 The goal

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 is not entirely found in the chosen strategy.

 the search for the current goal :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :6

 is finished in the chosen strategy

 The current goal to prouve is :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 for the region :8

%exactly thesame process will happen for region 8
%however, the data are more doubtful for the region
```

```
%so all the rules will fail, since a threshold exists

 The current rule under study is :

[noch_h(_10037)]

<---

[current_region(_10037),
 noch([true,_10037]),
 greater(cf(noch([true,_10037]),_10038),75)] 100

current_region(8) is verified
noch([true,8]) is verified
greater(cf(noch([true,8]),_10038),75)
 cannot be proved,the rule fails

 The current rule under study is :

[cc_h(_10038)]

<---

[current_region(_10038),ratio([true,_10038])] 100

current_region(8) is verified
ratio([true,8]) cannot be proved,the rule fails

 Everything which could be deduced was deduced.

 The goal

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 is not entirely found in the chosen strategy.

 the search for the current goal :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :8

 is finished in the chosen strategy

 The current goal to prouve is :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 for the region :9

 The current rule under study is :

[noch_h(_10375)]

<---
```

```
[current_region(_10375),
 noch([true,_10375]),
 greater(cf(noch([true,_10375]),_10376),75)] 100

current_region(9) is verified
noch([true,9]) is verified
greater(cf(noch([true,9]),90),75) is verified

Its premise [current_region(9),
             noch([true,9]),
             greater(cf(noch([true,9]),90),75)]

is verified; so its conclusions [noch_h(9)]
are entered in the bb

 The current rule under study is :

[cc_h(_12039)]

<---

[current_region(_12039),ratio([true,_12039])] 100

current_region(9) is verified
ratio([true,9]) is verified
Its premise [current_region(9),ratio([true,9])]

is verified; so its conclusions [cc_h(9)]
are entered in the bb

% a conflict will arise here since the blackboard now
% contains contradictory results for region 9

 the system found something but

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

is not entirely found.

 So another scanning of the knowledge-base is performed...

 The current rule under study is :

[noch_h(_13759)]

<---

[current_region(_13759),
 noch([true,_13759]),
 greater(cf(noch([true,_13759]),_13760),75)] 100

 The current rule under study is :

[cc_h(_13760)]

<---
```

```
[current_region(_13760),ratio([true,_13760])] 100

Everything which could be deduced was deduced.

% There are only two rules to fire and they were already
% fired so :

 The goal

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

 is not entirely found in the chosen strategy.

 the search for the current goal :

[fire_h(_2977),regen_h(_2977),cc_h(_2977),noch_h(_2977)]

for the region :9

 is finished in the chosen strategy

leaving the object level interpreter...

-------------------------------------------------------

the commands selected by the meta-level
are completed. Let's find another strategy...


 present content of the bb :
bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]])

bb(results_by_strategy,
   [[2],
    [[noch_h(4),[6,85,0]],
     [noch_h(5),[6,85,0]],
     [noch_h(6),[6,80,0]],
     [noch_h(9),[6,90,0]],
     [cc_h(9),[7,25,0]]]])

% The last strategy defined by the set of fired
% metas-rules : [2] found the above results

bb(intermediary_results,[[2],[]])

bb(agenda_strat,
   [[[4],
     [activ(separ_expert,reset,[4,5,6,8,9]),
      put_in_bb(data_base,[regions_of_study([4,5,6,8,9])])]],
    [[3,5],[activ(nochange_expert,reset,[[4,5,6,8]]),
            activ(ratioing_expert,reset,[9])]],
    [[6],[activ(nochange_expert,reset,[[9]])]],
    [[2],[met([pot_knb,4],knb,[[*],1]),
          demontre([4,5,6,8,9])]]])

bb(intermediary_results,
```

```
   [[6],[[noch([true,9]),nochange_expert,95,5]]])

bb(results_by_strategy,[[6],[]])

bb(intermediary_results,
   [[3,5],[[noch([true,4]),nochange_expert,90,5],
           [noch([true,5]),nochange_expert,90,5],
           [noch([true,6]),nochange_expert,90,10],
           [noch([true,8]),nochange_expert,90,20],
           [ratio([true,9]),ratioing_expert,35,10]]])

bb(results_by_strategy,[[3,5],[]])

bb(intermediary_results,
   [[4],[[separ([[4,5,6,8],[9]]),separ_expert,100,0]]])

bb(results_by_strategy,[[4],[]])

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])

bb(final_results,[[],[]])


 present content of the knowl. base :

knb([cc_h(_13830)],
    [current_region(_13830),ratio([true,_13830])],100,7)

knb([noch_h(_13830)],
    [current_region(_13830),noch([true,_13830]),
     greater(cf(noch([true,_13830]),_13831),75)],100,6)


 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current meta-rule is :
[call_arbitrator] <----- [dummy] 19

The mrule is verified;
 its conclusions are entered in the  agenda

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 execution of  the commands selected by the meta level :
[call_arbitrator]
////////////////////////////////////////////////////////////////
 entering the arbitrator...
let's find the best result for each region
looking for the final result for region :4
the choice is between :
[[fire_h(4),[unknown]],
 [regen_h(4),[unknown]],
 [cc_h(4),[unknown]],
 [noch_h(4),[[[2],6,85,0]]]]
 The best result is :[noch_h(4),[[[2],6,85,0]]]
looking for the final result for region :5
the choice is between :
[[fire_h(5),[unknown]],
 [regen_h(5),[unknown]],
 [cc_h(5),[unknown]],
 [noch_h(5),[[[2],6,85,0]]]]
 The best result is :[noch_h(5),[[[2],6,85,0]]]
looking for the final result for region :6
the choice is between :
[[fire_h(6),[unknown]],
 [regen_h(6),[unknown]],
 [cc_h(6),[unknown]],
 [noch_h(6),[[[2],6,80,0]]]]
 The best result is :[noch_h(6),[[[2],6,80,0]]]
looking for the final result for region :8
the choice is between :
[[fire_h(8),[unknown]],
 [regen_h(8),[unknown]],
 [cc_h(8),[unknown]],
 [noch_h(8),[unknown]]]
 The best result is :[region(8),[nothing_deduced]]
```

```
looking for the final result for region :9

the choice is between :

[[fire_h(9),[unknown]],
 [regen_h(9),[unknown]],
 [cc_h(9),[[[2],7,25,0]]],
 [noch_h(9),[[[2],6,90,0]]]]

%resolution of the conflict...

 The best result is :[noch_h(9),[[[2],6,90,0]]]

% Since it has the higher confidence measure

leaving the arbitrator...

////////////////////////////////////////////////

the commands selected by the meta-level
are completed. Let's find another strategy...


present content of the bb :

bb(final_results,
   [[],[[noch_h(4),[[[2],6,85,0]]],
        [noch_h(5),[[[2],6,85,0]]],
        [noch_h(6),[[[2],6,80,0]]],
        [region(8),[nothing_deduced]],
        [noch_h(9),[[[2],6,90,0]]]]])

% These are the final results found by the last
% strategy

bb(results_by_strategy,[[1],[]])

bb(intermediary_results,[[1],[]])

bb(agenda_strat,
   [[[4],[activ(separ-expert,reset,[4,5,6,8,9]),
          put_in_bb(data_base,
                    [regions_of_study([4,5,6,8,9])])]],
    [[3,5],[activ(nochange_expert,reset,[[4,5,6,8]]),
            activ(ratioing_expert,reset,[9])]],
    [[6],[activ(nochange_expert,reset,[[9]])]],
    [[2],[met([pot_knb,4],knb,[[*],1]),
          demontre([4,5,6,8,9])]],
    [[1],[call_arbitrator]]])

bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]])

bb(results_by_strategy,
   [[2],[[noch_h(4),[6,85,0]],[noch_h(5),[6,85,0]],
         [noch_h(6),[6,80,0]],[noch_h(9),[6,90,0]],
         [cc_h(9),[7,25,0]]]])
```

```
bb(intermediary_results,[[2],[]])

bb(intermediary_results,
   [[6],[[noch([true,9]),nochange_expert,95,5]]])

bb(results_by_strategy,[[6],[]])

bb(intermediary_results,
   [[3,5],[[noch([true,4]),nochange_expert,90,5],
           [noch([true,5]),nochange_expert,90,5],
           [noch([true,6]),nochange_expert,90,10],
           [noch([true,8]),nochange_expert,90,20],
           [ratio([true,9]),ratioing_expert,35,10]]])

bb(results_by_strategy,[[3,5],[]])

bb(intermediary_results,
   [[4],[[separ([[4,5,6,8],[9]]),separ_expert,100,0]]])

bb(results_by_strategy,[[4],[]])

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])


 present content of the knowl. base :


 preprocessing for the beginning of a cycle...

preprocessing for the beginning of a cycle is completed

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 consulting the meta-level...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

entering the meta-level interpreter...

the current meta-rule is :
[send_message] <----- [dummy] 20

% The results should be sent to a file which would be
% read by the manager. Here, since the manager is the
% user, no transfer is done

 The mrule is verified;
 its conclusions are entered in the agenda

leaving the meta-level interpreter...

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 the meta level is consulted
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 execution of  the commands selected by the meta level :

[send_message]

% send-message sends the present message

This is the end of the session for the node...

% to the terminal

 final content of the bb :

bb(results_by_strategy,[[7],[]])

bb(intermediary_results,[[7],[]])

bb(final_results,
   [[],[[noch_h(4),[[[2],6,85,0]]],
        [noch_h(5),[[[2],6,85,0]]],
        [noch_h(6),[[[2],6,80,0]]],
        [region(8),[nothing_deduced]],
        [noch_h(9),[[[2],6,90,0]]]]])

bb(results_by_strategy,[[1],[]])

bb(intermediary_results,[[1],[]])

bb(agenda_strat,
   [[[4],[activ(separ_expert,reset,[4,5,6,8,9]),
          put_in_bb(data_base,
                    [regions_of_study([4,5,6,8,9])])]],
    [[3,5],[activ(nochange_expert,reset,[[4,5,6,8]]),
            activ(ratioing_expert,reset,[9])]],
    [[6],[activ(nochange_expert,reset,[[9]])]],
    [[2],[met([pot_knb,4],knb,[[*],1]),
          demontre([4,5,6,8,9])]],
    [[1],[call_arbitrator]]])

bb(data_base,[[],[[regions_of_study([4,5,6,8,9])]]])

bb(results_by_strategy,
   [[2],[[noch_h(4),[6,85,0]],[noch_h(5),[6,85,0]],
         [noch_h(6),[6,80,0]],[noch_h(9),[6,90,0]],
         [cc_h(9),[7,25,0]]]])

bb(intermediary_results,[[2],[]])

bb(intermediary_results,
   [[6],[[noch([true,9]),nochange_expert,95,5]]])

bb(results_by_strategy,[[6],[]])

bb(intermediary_results,
   [[3,5],[[noch([true,4]),nochange_expert,90,5],
```

```
                [noch([true,5]),nochange_expert,90,5],
                [noch([true,6]),nochange_expert,90,10],
                [noch([true,8]),nochange_expert,90,20],
                [ratio([true,9]),ratioing_expert,35,10]]])

bb(results_by_strategy,[[3,5],[]])

bb(intermediary_results,
   [[4],[[separ([[4,5,6,8],[9]]),separ_expert,100,0]]])

bb(results_by_strategy,[[4],[]])

bb(agenda_upper,[[],[detect_ch(exhaustif,[4,5,6,8,9])]])


leaving the change-detection expert node...

% This is the end of the session...
************************************************************
```

APPENDIX B

THE EXPLANATION FACILITY

B.1 FUNCTIONAL DESCRIPTION

B.1.1 Program Description

The facility is contained in 4 modules :

1. module : just (in just.pro)

2. module : justwhy (in justwhy.pro)

3. module : justhow (in justhow.pro)

4. module : procsec (in procsec.pro)

The explanation facility also include files. One is to be instantiated for the different existing nodes. By consulting it, the facility learns which node exists and which file related to each node should be consulted. That file is "storedata.dat". It must exist and be instantiated. The other files are node- dependent. Once a node is selected by the user, the facility has to consult :

1. the related knowledge-base

2. the related meta-knowledge base

3. the file containing "secondary" data providing several characteristics of the selected node

4. the file created by the justifier-interface of the selected node in a previous consultation session.

B.1.2 Module Description

The module "just" contains a goal which is the predicate "explainfacility". When activating the explanation facility, this predicate is verified.

This module checks the validity of the node name entered, the validity of the request and dispatch the request to "justwhy" or "justhow", which are the files which handle the two types of questions.

The module "procsec" contains secondary procedures which are common with the expert system shell. This module is not presented here.

The module "justhow" handles how-type questions. It prompts for the rule number or the element name, then tries to retrieve relevant information. If found, the information is displayed in a formatted way, otherwise, the system answers that the element or rule was not used during the consultation session.

The same procedure is used for why type questions.

B.1.3 Session Description

The session is interactive and the user is prompted for prerequisites :

1. the node involved.

2. the type of question the user wants to ask (how type or why type questions).

3. the kind of object to explain : an element or a rule.

4. If an element is requested, the program prompts for its name

5. If a rule is requested, the user is asked if the rule belongs to the object or to the meta-level,

6. and then the rule number.

The system provides stored information about the request and prompts for another request.

B.2 PROCEDURE FOR USE

B.2.1 Presentation Of The Requested Input

Here is the beginning of a session:

The answers are between <>, and what may be entered is in ""
:

```
 Welcome to the explanation facility
 1) give me the name of the node you want explanations
about:
that node may be :
          changeexpert
          separexpert
 Enter any of the listed names :
< enter "changeexpert" or "separexpert" >
```

If you want explanations about HOW an element or a rule was used reply by (h)ow .

If you want explanations about WHY an element or a rule was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n) ?

< "h" or "w" or "n" /* here, "h" is entered */ >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ? :

< "r" or "e" /* here, "r" was entered */ >

If you want explanations about an object-rule, type o,
If you want explanations about a meta-rule, type m,
your choice :

< "o" or "m" /* here, "o" was entered ">

Which rule ? (enter the number followed by a '.')

< a number followed by a "."°/* here "1." was entered */ >

B.2.2 Presentation Of The Output

The presentation of the output is done via examples. The consultation begun in the previous paragraph is followed.

```
The rule1 : noch_h(X) <--- current_region(X),
                           noch([X,true]),
                           greater(cf(noch([true,X]),_1005),75)
```

was triggered to help in the deduction of :

noch_h(4)

because

current_region(4) was given as input data

noch([4,true])was found by
the sub-node "nochange-expert"

with a MB =90 and a MD =05

the condition greater(cf(cva,85),75)
was verified

and the final measures of belief and disbelief

for noch_h(4) were:

MB = 85
MD = 00

and were found in context of the strategy defined
by the set of rules: [2]

in firing the object-rule 6

The rule1 : noch_h(X) <--- current_region(X),
noch([X,true]),
greater(cf(noch([true,X]),_1005),50)

was triggered to help in the deduction of :

noch_h(5)

because

current_region(5) was given as input data

noch([5,true])was found by
the sub-node "nochange-expert"

with a MB =90 and a MD =05

the condition greater(cf(cva,85),75)
was verified

and the final measures of belief and disbelief
for noch_h(5) were:

MB = 85
MD = 00

and were found in context of the strategy defined
by the set of rules: [2]

in firing the object-rule 6

The rule1 : noch_h(X) <--- current_region(X),
noch([X,true]),
greater(cf(noch([true,X]),_1005),50)

was triggered to help in the deduction of :

noch_h(6)

because

current_region(6) was given as input data

noch([6,true])was found by
the sub-node "nochange-expert"

with a MB =90 and a MD =10

the condition greater(cf(cva,80),75)
was verified

and the final measures of belief and disbelief
for noch_h(6) were:

MB = 80
MD = 00

and were found in context of the strategy defined
by the set of rules: [2]

in firing the object-rule 6

The rule1 : noch_h(X) <--- current_region(X),
noch([X,true]),
greater(cf(noch([true,X]),_1005),50)

was triggered to help in the deduction of :

noch_h(9)

because

current_region(9) was given as input data

noch([9,true])was found by
the sub-node "nochange-expert"

with a MB =90 and a MD =0

the condition greater(cf(cva,90),50)
was verified

and the final measures of belief and disbelief for noch_h(9) were:

MB = 90
MD = 0

and were found in context of the strategy defined

```
by the set of rules: [2]

in firing the object-rule 6


If you want explanations about HOW an element or a rule

was used reply by (h)ow .

If you want explanations about WHY an element or a rule

was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n) ?

< "h" >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ? :

< "r" >

 If you want explanations about an object-rule, type o,
 If you want explanations about a meta-rule, type m,
 your choice :

< "m" >

Which rule ? (enter the number followed by a '.')

< "2." >

The rule2 : [met([pot_knb,4],knb,[[*],1]),
           demontre(_2683)]

              <---

           [command-upper([detect_ch(exhaustif,_2683)])]
 Prio : 18

was fired because :

     command-upper([detectect_ch(exhaustif,[4,5,6,8,9])])

was verified  and
```

helped in deducing :

noch_h(4) by use of the rule : 6
noch_h(5) by use of the rule : 6
noch_h(6) by use of the rule : 6
noch_h(9) by use of the rule : 6
cc_h(9) by use of the rule : 7

If you want explanations about HOW an element or a rule was used reply by (h)ow .

If you want explanations about WHY an element or a rule was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n) ?

< "h" >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ? :

< "e" >

Which (sub-)goal ? :

< noch([4,true]). >

In the context of the strategy defined
by the set of rules : [3,5]

noch([4,true]) was given as data

as the result of the sub-node "nochange-expert"

If you want explanations about HOW an element or a rule was used reply by (h)ow .

If you want explanations about WHY an element or a rule

```
was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n) ?

< "h" >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ? :

< "e" >

Which (sub-)goal ? :

< noch_h(4). >

In the context of the strategy defined
by the set of meta-rules: [2],

     noch_h(4) was found by use of the rule(s) :

     [noch_h(X)] <---- [current-region(X),
                        noch([X,true]),
                        greater(cf(noch([X,true]),_23),75)]

                  with a CF of 100

 and the final best result for the (sub-)goal noch_h(4)

 was found

 *  by the strategy :[2]

 the last fired rule 6

 found the result with a measure of belief of : 85
 and a measure of disbelief of                : 0


 If you want explanations about HOW an element or a rule

 was used reply by (h)ow .

 If you want explanations about WHY an element or a rule
```

was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n) ?

< "w" >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ? :

< "e" >

Which element ?

< "noch h(4)" >

noch_h(4) is a goal and you may have asked to prove it...

If you want explanations about HOW an element or a rule

was used reply by (h)ow .

If you want explanations about WHY an element or a rule

was used reply by (w)hy .

If you do not want explanations reply by (n)o :

Your answer (h/w/n)?

< "w" >

If you want explanations for a rule reply by (r)ule .

If you want explanations for an element reply by (e)l. .

Your answer (r/e) ?:

< "e" >

Which element ?

< "noch([4,true])." >

The element noch([4,true]) with

```
        a measure of belief of 90
      and a measure of disbelief of 05

helped in the deduction of

        noch_h(4)

in the context of the strategy defined
by the set of meta-rules: [2]

by firing of the object-level rule 6:

        noch_h(X)  <--- [  current-region(X),
                           noch([X,true]),
                           greater(noch([X,true],_345),75) ]

                    with a CF of 100


because:

        current-region(4) was verified
        greater(noch([4,true],85),75) was verified


If you want explanations about HOW an element or a rule

was used reply by (h)ow .


If you want explanations about WHY an element or a rule

was used reply by (w)hy .


If you do not want explanations reply by (n)o :


Your answer (h/w/n) ?

< "n" >

Au revoir !
```

APPENDIX C

THE KNOWLEDGE ACQUISITION SYSTEM

C.1 A SESSION WITH THE KNOWLEDGE ACQUISITION SYSTEM

The proposed example follows the one used in the documentation about the consultation part of the expert system shell.

We suppose the user wants to enter a new rule concluding about the non-existence of a change and we also suppose the system has access to th following models :

(N.B. : comments are preceeded by a '%')

```
% the knowledge-base of the node "change-expert" is :
% see the documentation of the consultation part of the shell


                        *****************
                        *               *
 $ NAME                 *     chknB     *
                        *               *
                        *****************

*****************************************************

Predicate:    num_rule

Description:  provides the actual number of
                rules contained
              in the set. It is updated by the KAS.

*****************************************************

/* gives the actual number of knB rules */


num_rule( 7 ) .

*****************************************************

Predicate:    knB

Description:  only one predicate currently available to
              embody the rules. The format is :
              [conclusions],[premise],CF,rule number

*****************************************************


/* the knowledge base */

pot_knB([noch_h(X)],
        [current_region(X),
       noch([true,X]),
       greater(cf(noch([true,X]),CF),75)],
        100,6).

pot_knB([cc_h(X)],
     [current_region(X),ratio([true,X])],100,7).

%the knowledge-base thus contains only two rules
% though the given number of rules is 7. This may
%happen after changes in the knowledge-base, such as
% deletions.
```

```
% the format of the rule and the meaning of the elements
% is explained in the previous example

% First model :

/* model for the rules deducing the
existence of some goal (positive or negative);
this model is a root model */

/* set of rules described by the model */

example(chmodgoal,[6,7]).

% the set of rules represented by the model. The first
% argument is the model name, the second, the set

/* set of typical actions performed by the rules */

typic_action(chmodgoal,[ [cc_h,[X],1],
                         [noch_h,[X],1]]).

% the set of typical actions. The list in second argument
% contains those actions :
% format of a described action :
%          [<predicate of the conclusion>,
%           [the argument(s) of the conclusion],
%           weight for the conclusion ]

% computation of the weight (example) :
% the first element cc_h has a weight of 1 because
% it is present in only one rule which certainty
% factor is 1.


/* set of correlations existing on the actions */

correl_action(chmodgoal,[]).

% no correlation is known for the action part
% of the rules

/* set of typical premises exsting on the rules */

typic_premise(chmodgoal,[[current_region,[X1],2],
                         [noch,[[true,X2]],1],
                        [greater,[G1,G2],1],
                        [ratio,[[true,X3]],1]]).

% set of typical premise elements
% the notation is the same as above. The variables
% in the set of arguments of the elements are given
% different names because otherwise, they would imply
% a correlation among the elements
```

```
/* set of correlations existing on the rules */

correl_premise(chmodgoal,[]).

%no correlation exists for the premise elements


/* pointer to more general model */

more_genl(chmodgoal,[]).

% the model is a root

/* pointer to more specific model */

more_spec(chmodgoal,[chmodgopo,chmodgone]).

% two more specific models are pointed . Among them,
% only the first one, which characterizes the rules
% which conclude positively about goals (for the node)
% actually exists and is now displaied :


% Second model

/* model for the rules deducing
the existence of some goal (positive or
negative); this model is a root model */

/* set of rules described by the model */

example(chmodgopo,[6,7]).

/* set of typical actions performed by the rules */

typic_action(chmodgopo,[ [cc_h,[X1],2],
                         [noch_h,[X2],2]]).

/* set of correlations existing on the actions */

correl_action(chmodgopo,[]).

/* set of typical premises exsting on the rules */

typic_premise(chmodgopo,[[current_region,[X1],2],
                         [noch,[[true,X2]],1],
                         [greater,[G1,G2],1],
                         [ratio,[[true,X3]],1]]).


/* set of correlations existing on the rules */

correl_premise(chmodgopo,[[[noch,[[true,X]]],
                          [greater,[G1,X]],1]]).
```

```
% a correlation exists between the element noch
% with one argument of the type [true,X], and
% the element greater with one argument of the
% type [Gl,X]
% The correlation has a weight of 1
%
/* pointer to more general model */

more_genl(chmodgopo,[chmodgoal]).


/* pointer to more specific model */

more_spec(chmodgopo,[chmodfire,chmodnoch]).

%the pointed models does not exist actually and they
% thus will fail to fit the entered rule. The
% system will backtrack to their father, the second
% model (see below)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% an example of a session with the
% knowedge acquisition system:


Welcome to the object-level knowledge acquisition system.

1) give me the name of a node you want to update ;

that node may be :

          change_expert
          separ_expert


%the only existing nodes are the one
%displayed above

Enter any of the listed names :

%the user entered: <change_expert.>

Thank you...

Please,
Enter an object-rule with the following format :

knB(<List_of_Actions> <List_of_premises> <CF> <Rule_Number> ;

You are prompted for the relevant data ;

The current and only available functor is knb
```

```
Enter the actions to perform :

Format: [<action1>,<ac...>]
finish by a '.'

%the user entered : <[noch_h(X)].>

The action set is :[noch_h(X)]

Enter a premise one by line ; Finish by a "."

%the user entered : <a.>
%'a' is a new element unknown to the system

Enter a premise one by line ; Finish by a "."

%the user entered : <noch([true,X]).>

Enter a premise one by line ; Finish by a "."

%the user entered <..>

The set of premise_elements is :

a noch([true,X])

The element a has not been recognized as

a standard element ;

No rule contains a
in its premise ;

Please
confirm or abort (answer by c or a) :

If the element is a condition, you have to enter its meaning
separately; in this case, the file containing the conditions
definitions has to be updated ;

If it is an element, the set of premise- element
will be updated :

%the user confirms th element 'a'
% and enter <c>

If a is a condition, enter c ;
If a is an element , enter e :

%the user described the element 'a' as an
% element and entered <e>

The set 'premise_obj' includes now the element

% the system will know from now on that
% is a plausible element of a premisse
```

Enter the certainty factor (between 0 and 100) :

%the user entered the certainty factor : <100.>

%Now the system finds the best fitting model which
%is the second one.

The rules concluding on the same subject often have

the following functor :

current_region

This is concluded with a weight of 2

Do you want to add such a premise

(answer by y or n) ? :

%the user entered <y>
%let's recall that this element is useful in
%instantiating a rule to the region number

Enter a premise one by line ; Finish by a "."

%the user entered : <current_region(X).>

Enter a premise one by line ; Finish by a "."

%the user entered <..>

The rules concluding on the same subject often have

the following functor :

greater

This is concluded with a weight of 1

Do you want to add such a premise

(answer by y or n) ? :

% answer: <n>
% the user did not find the element of interest

The rules concluding on the same subject often have

the following functor :

ratio

This is concluded with a weight of 1

Do you want to add such a premise

```
(answer by y or n) ? :

%answer <n>

The rules concluding on the subject often have

the following premises together :

[noch,[[true,X]]] [greater,[_2453,X]] 1

Do you want to create such a correlation ?

(answer by y or n) :

% answer: <y>
% Finally, the user found that it may be useful
% to put a threshold on the confidence of the
% result noch([true,X])

Propose a premise . Finish by a . :

%entry: <greater(cf(noch([true,X]),Y),75).>

Propose a premise . Finish by a . :

%answer: <..>

The rule number is 8 as well as the total number of rules

% the system updates then the knowledge-base
% This is the end of the session
%
% Consulting the new knowledge base of the node
% would display the following content :


num_rule(8).

pot_knb([noch_h(_3004)],
      [current_region(_3004),
       noch([true,_3004]),
         greater(cf(noch([true,_3004]),_3005),75)],
      100,6).

pot_knb([cc_h(_3004)],
      [current_region(_3004),
       ratio([true,_3004])],100,7).

% plus the new rule :
% Let's recall that the order of the rule at the
% object level is not important

pot_knb([noch_h(_3004)],
      [a,
       noch([true,_3004]),
```

```
current_region(_3004),
greater(cf(noch([true,_3004]),_3005),75)],
100,8).
```