PROLOG FOR EXPERT SYSTEMS : a case study

Luis Moniz Pereira
Eugenio Oliveira

Departamento de Informatica
Universidade Nova de Lisboa
Quinta da Torre
2825 Monte da Caparica
Portugal

September 1983

Abstract

We describe a computer system , interrogatable in a flexible subset of Portuguese, and implemented in Prolog on a small machine, which embodies and assimilates expert knowledge on environmental biophysical resources evaluation, is capable of explaining the application of that knowledge to a territorial database, and also of answering questions about its linguistics abilities. The system was developed in two man-years, under contract with the Portuguese Department of the Environment. Finally, we adduce three sets of reasons regarding the suitability of the Prolog language for expert systems implementation.

Keywords: Prolog, Logic Programming, Expert Systems, Natural Language.

Contents

1. Knowledge engineering

Arisen within the Artificial Intelligence field, intelligent computer programs embodying, accepting and using organized knowledge concerning one or several specific human expertise areas are known as Knowledge-Based Systems or Expert Systems (ES). The aim of an ES is to help or to partially substitute in an advantageous way, experts in their knowledge domains.

Computer scientists who build up such systems by transferring human knowledge to the program and developing the necessary tools to satisfactorily perform an expert task are working in Knowledge Engineering.

Basically, an ES consists of two fundamental components : Knowledge plus Control. Knowledge is captured in a Knowledge Base and can be represented in a number of forms ranging from table relations and predicate calculus to production rules or frames. A further distinction could separate out a traditional Data Base composed of simple facts, the extensive part (which could be pre-existent), and a deductive component (the intensive part) representing more complex expert knowledge acting as elaborate laws (the Knowledge Base).

Control is enacted by an inference engine, envolving some appropriate knowledge acquisition, deductive and explanatory strategies and heuristics, both general and peculiar to the particular problem domain, plus the planning necessary to accomplish efficient Knowledge Base access.

Well known and broadly accepted examples of Expert Systems have been implemented in Lisp or in some Lisp-like language, as it is the case of Mycin for diagnosis of blood infections, and Prospector, for mineral exploration, among many others.

The principal attractive features of an Expert System are :

- Deduction over Data Base information
- Explanations
- Selfknowledge

and at least for some of them

- Interactive knowledge acquisition
- Natural language interface
- Fuzzy reasoning

Research is mostly pursued within the following topics:

- Knowledge representation
- Knowledge acquisition
- Inference mechanisms
- Natural language

and, which emcompasses the first three of these aspects, on domain independent systems implementation.

Expert Systems' widespread use can provide abundant and cheap intellectual power for the following types of uses and examples :

- Diagnosis (of diseases)
- Evaluation (of resources)
- Configuration (of computer systems)
- Consultation (of knowledge bases)
- Generation (of programs)
- Design (of products)
- Maintenance (of artifacts)
- Teaching (of subject matters)

The use of ES to accumulate and transfer organized expert knowledge could be an extremely important side-effect of actual work in this field. More

generalized frameworks aiming to guide experts in multiple domains would be of great help.

2. ORBI an expert system in Prolog

ORBI [OLI], [PE1], (a Portuguese acronym for biophysical resource evaluation) is an Expert System whose domain doesn't refer to a single discipline but encodes multidisciplinary knowledge of several experts in domains as diverse as hydrology, geology, fauna, flora and climate, among others, enabling ORBI to make judgements over an integrated knowledge expertise .

Our ES comprises a data base of facts and a knowledge base of inference rules, and is capable of answering queries by using the rules to reasoning, about the facts. It also explains its reasoning. in more or less detail, according to the needs of the user. Rules and facts can be updated by the user, new items being automatically incorporated without any further programming.

Queries are expressed in natural language (Portuguese). They can be about the domain of expertise, including how the knowledge is organized, or about the system's natural language competence, thereby alleviating the user from the need to consult the manual.

2.1 Domain of expertise

The domain of expertise concerns the evaluation of the biological and physical offer of the environment. The typical region size is that of the county, which is described by means of 23 descriptors. To each descriptor there corresponds a map of the region, which is covered by a uniform grid forming squares with 200 meters side. On each square the value of descriptors is digitized to integers ranging from 1 to 5, with some exceptions. Furthermore, to each value is associated a representativity, an integer ranging from 1 to 5, describing the extent to which that value is homogeneous in the square. A county typically comprises some 6.000 squares. Examples of descriptors are:

d1 — microclimate quality
d9 — risks of soil erosion
d11 — vegetation balance.

Given the observed data base of ground facts as expressed by descriptors, more abstract derived information can be obtained by defining factors and aptitudes. A factor is the result of a combination of descriptors. It has an integer value ranging from 1 to 5, or the value s, meaning it is a selective factor. Again, to each value is associated a representativity ranging from 1 to 5. The value of a factor is computed from the values of the underlying descriptors through a rule provided by experts. A rule may have several alternative cases, depending on the actual values of the descriptors. For example, a descriptor may have a value that overides all other descriptors and then completely defines the factor by itself, or the value of the factor may be a weighed sum of all or some of the underlying descriptors, according to the actual values present. The representativity of the factor value however, is always a weighed sum of the representativities of the intervening descriptors. There are 16 factors. Examples are :

f1 — climatic comfort
f3 — resistance of soil to erosion

f6b -- cinegetic and landscape riches.

Aptitudes are likewise defined in terms of factors, as these are in terms of descriptors, through inference rules supplied by a team of experts. There exist 5 aptitudes :

a1 -- lodging
a2 -- non-intensive agriculture
a3 -- intensive agriculture
a4 -- industry
a5 -- recreation.

Finally, a biophysical synthesis rule relies on the aptitude values and representativities to extract the more promising areas according to development objectives, and a nature preservation rule advises protection measures.

It took approximately two man-years to develop and implement ORBI using a DEC PDP11/23 with 128K under the RT11 Operating System.

The knowledge of the experts is further made explicit by a large collection of explanatory texts, one for each possible value of each descriptor.

Protocol (translated)

-which is the rule for aptitude 1 ?

```
a1=0  if        f4=4   or  f4=5
            or
               f6a=s  or  f6a=3
            or
               f6b\=0  and  f6b\=e  and  f6b\=f
            or
               f6c=5   or  f6f=5

      else     a1=(40*f1+25*f2+20*f2a+5*f2b+5*f3+5*f5)/100.
```

-what is the value of aptitude 1 at point 97,699 ?

aptitude 1 - lodging - is apt (3 with representativity 4) because :
3 was computed from (40*2+25*2+20*5+5*4+5*4+5*2)/100
by substitution in (40*f1+25*f2+20*f2a+5*f2b+5*f3+5*f5)/100.

factor 1 - climatic comfort - is 2 (with representativity 3) because :
2 was computed from (60*2+40*4)/10
by substitution in (60*d1+40*d2)/10.

descriptor 1 - microclimate quality - is 2 (with representativity 2) since the potential microclimate quality for permanent human settling is low because it is a steep-sloped region, facing north and the dominant winds

descriptor 2 - homogeneous climatic zones - is 4 (with representativity 5)

because the whole of the thermo-pluviometric characteristics indicate conditions favourable to permanent human settlement

factor 2 – aptness of soil for foundations – is 2 (with representativity 3) because : 2 was computed from (65*1+15*5+10*5+10*3)/100
by substitution in (65*d4+15*d5a+10*d3+10*d5)/100.

descriptor 4 – mechanical resistence of the substract – is 1 (with representativity 2) on account of chalk formations

descriptor 5a – the slope steepness – is 5 (with representativity 5) because the average slope is between plane and moderate

descriptor 3 – permeability of the substract – is 5 (with representativity 4) because of dune formations

descriptor 5 – geodynamic stability of the substract – is 3 (with representativity 4) because of dune formations

factor 2a – potential in supply water – is 5 (with representativity 5) because:

descriptor 8 – potential in subterranean waters – is 5 (with representativity 5) because of easily explorable captive Mio-Pliocenic resources

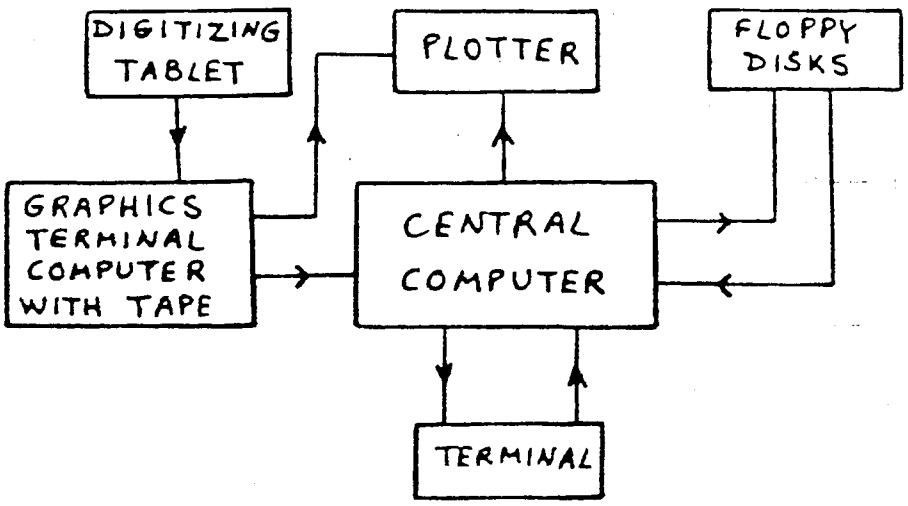
factor 2b – resistence to polution of the aquiferous layer – is 4 (with representatity 5) because :

descriptor 9 – vulnerability to polution of aquiferous layers – is 4 (with representativity 5) because the most importantly occurring aquiferous present a high protection value since they belong to the captive Mio-Pliocenic layer

factor 3 – resistence of soil to erosion – is 4 (with representativity 5)

etc.

2.2 System configuration



2.3 Knowledge representation

Edinburgh Prolog notation is used henceforth. Facts, the observed points characteristics, are Represented by a set of unit clauses of the form

```
point(X-Y,d(v1-r1,v2-r2,...)).
```

where X and Y are the point coordinates and d is the descriptors functor containing as arguments all value-representativity pairs, one for each descriptor at that point. Points are digitezed from maps and kept in the Data Base as they were records with two fields :

```
-- coordinates : integer
-- descriptors : array of pairs of integers
```

We also have stored as facts, a set of justifying messages, one for each possible value of each descriptor.

But far more important is the expertise assembled and organized as inference rules enabling ORBI to compute factors from descriptors, aptitudes from factors and the biophysical synthesis from aptitudes and the nature preservation rule. Rules are pieces of modular knowledge that can be dealed with separately, whether being deleted, asserted, modified, displayed or evaluated. Such rules, production rules with a situation part and an action part, are quite naturally expressed as Prolog non-unit clauses. These are made clearer understandable by means of pre-declarated operators (and, or, if, else, not...).

Clause heads, naming a specific concept (aptitude, factor, ...), receive all necessary data base information in argument D and produce the final inferred value and representativity in its arguments V-R.

An example rule for the "lodging" aptitude 1 is input as :

```
aptitude n.1
name : lodging
```

and them the very same form as we can see in the above output protocol.

This rule is then 'compiled' into a prolog clause (which can be 'uncompiled' into the original input notation).

We have defined a rather general user-friendly language for rule definition which allows the user to impart structure to rules. Both the explanation and the planning modules recognize this structure. Consequently, explanations comply to the original rule structure as expressed by the user, and the

planner gathers from the structure hints on the query evaluation optimization.

The 'compilation' of rules into Prolog allows for automatic generation of goals implicit in the user's rule (such as the goals necessary for the computation of the plausibility of inexact reasoning), and consequently the external rule notation is cleaner.

2.4 Execution process

As the Data Base of points is too large to reside in central memory, access to secondary memory is necessary. In this application we partitioned the data base of facts (points) into many smaller files whose names express the range of point coordinates they comprise. Since a query is always about known points, ORBI only transfers to memory the minimum required files.

Deduction is carried out by chaining several rules, from descriptors values to the desired answers by backward chaining (with backtracking) through Prolog interpreter. Queries are presented as top goals to be derived by the interpreter through its own strategy (depth first, left to right). Consequently, what has to be build in an ES implemented in other languages (as Lisp), it is already done by prolog clause interpreter.

Anyway it is always possible to add some heuristics due to the problem as it is the case with ORBI. Prolog is particular well suited for writing simple interpreters which can act over clauses to be evaluated, transforming or delaying th

In ORBI, a small interpreter executes first all deterministic goals and sub-goals of the query and, on the fly, generates the deterministic part of the explanation tree containing variables standing for the non-determinate explanation parts. The objective is to achieve efficiency in queries envolving a set of solutions, because they differ only in their nondeterministic portions. Next, in the evaluation stage, the nondeterministic parts of the query are evaluated and the corresponding explanation structures are generated and bound to the variable slots in the overall explanation structure. The tips of this structure are the messages explaining the meaning of descriptor values. Finally, the results are presented and explained.

2.5 Selfknowledge

It is important for the required domain knowledge to be not only both well organized and efficiently used but also for it to be retrievable in a simple and understandable way to the user interface. ORBI keeps all its domain concepts and their multiple relationships, with other useful information, in a Metaknowledge module (knowledge about the knowledge domain).

It is organized as a three layer module as follows :

-- Templates, encapsulating the general categories involved and its known characteristics, as for exemple :

```
concept( quantity, instantiations, dependencies, contributions, meaning ).
```

-- the Schema, relating the concepts in an associative network For example :

```
factor( 16, tfact(NU,NA,DE,CO,ME), [descriptor], [aptitude,synthesis],
                                                          sig(fact) ).
```

stating that there are 16 factors, which depend on descriptors (whichever they were), and contributing to some aptitudes. The meaning of a factor as a general domain concept is kept in a "slot", as the argument of the functor 'sig'. The second argument of the unit clause indexes a node in the lowest level network where the factor's instantiations are.

-- Concrete entities. For example 'Non Intensive Agriculture' , one. of the aptitudes, (numbered 3) :

```
tapt( 3, 'Non Intensive Agriculture', [f1,f2a,f2b,f3], [s1], sig(a3) ).
```

This easily accessible structure together with a complex procedure responsible for looking into the rules and presenting them in a friendly format are quite adequate to instruct the user about what knowledge ORBI contains.

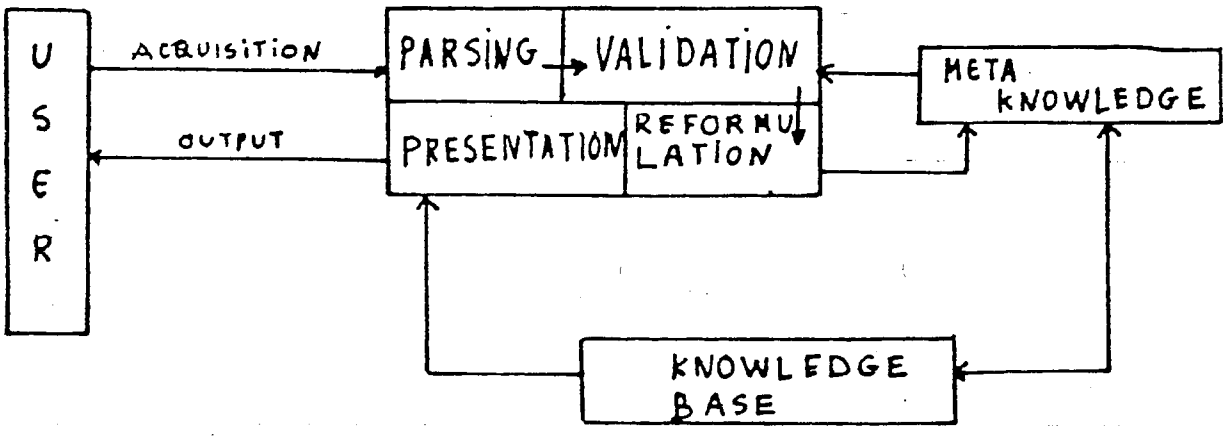
Once again our implementation language facilitates the task, because procedural code, for instance rules stated as non-unit Horn clauses, can be seen and manipulated as data by other procedures which inspect them.

We have strived not to duplicate any information in the knowledge base, so as to facilitate automatic updating of the relevant structures, and strived as well to have data structures that can be both easily read and used by other program parts, and being also easily executable program parts on their own.

2.6 Knowledge acquisition

Knowledge Bases often become obsolete, incomplete or erroneous. Domain expertise is a dynamic body of knowledge evolving in time. Thus, it is crucial to give the expert the possibility of interacting with the system and to instruct it about domain knowledge changes.

Knowledge acquisition is accomplished in ORBI in a natural and safe way by the cooperation of two modules : Acquisition and Metaknowledge.



New rules are entered in a simple and flexible input language according to displayed BNF grammar rules. Metaknowledge is consulted for the sake of syntactic and semantic consistency. A parser takes the new rule, input

as a single term, recognizes its concepts, its attributes and values, operators and functors, and translates them into a procedural form by building up an appropriate non-unit clause, acceptable to the Metaknowledge module.

We can sequence the acquisition process as follows :

-- consulting metaknowledge to instruct the expert about concept relations
-- displaying BNF grammar rules
-- entering the rule
-- parsing and checking it
-- showing the new rule
-- constructing the internal representation of the rule
-- redefining metaknowledge (if changes are definite)

A new Knowledge Base file is created with all improvements, which supersede old knowledge in subsequent consultations.

New facts can be entered off-line (digitized from maps for instance) and a separate program reconfigurates them as appropriate unit clauses in the Data Base.

3. Natural language competence

linguistic competence

ORBI's linguistic competence is achieved by means of a lexical and a syntactic-semantic analysis, transforming a natural language sentence into a directly evaluable list of Prolog goals. For example, the query :

Which are the points having the same value for the aptitude intensive agriculture?

will be transformed into the Prolog expression (cf. [PE3] for a definition of 'all') :

```
all ( V-S ,
      all ( [point:(X,Y)] , (p(X,Y,D),a(3,D,V-R)) same V , S ),
       L  ).
```

where 3 is the code for 'intensive agriculture', and
L is the answer in the form of a list of pairs V-S, such that for each value V of aptitude 3, S is the list of points with coordinates X,Y that have the value V

the lexical analysis

Like MICROSIAL [PIQ], our lexical analysis replaces each word of the input

sentence by its corresponding lexical category. That is done by indexing on the word entries of ORBI's dictionary.

the syntactic analysis

The syntactic-semantic analysis is realized by means of a core grammar containing context-free and sensitive rules (expressed in the definite clauses grammar formalism, see [PER]) with syntactic and semantic checks. These rules handle the fundamental structures of Portuguese, namely :

- yes/no questions;
- wh questions;
- commands;
- affirmative, negative, relative, prepositional, coordinate, extraposed and elliptic clauses;
- complex noun complementation, adjunction and abbreviations;
- universal, existential, numeral, definite and indefinite determiners;
- adjectives, verbs and adverbs;
- metalinguistic terminology (allowing the asking of questions about ORBI's linguistic competence itself);

This core grammar is independent of its application and transportable to other domains. It has indeed been done [PE2].

The core grammar is completed by a domain dependent module (10 times smaller than the core) containing structures, vocabulary (noun and verb phrases essentially) and semantic controls pertaining to ORBI's domain of expertise.

We have devoted a lot of energy to incorporate elliptic and extraposed structures in ORBI's linguistic competence, because they are essential to a good language interaction. A natural language interface unable to understand sentences involving these structures is not truly natural.

Syntactic and semantic controls verify number and gender agreements, designations of complex entities, and compatibilty between nouns and verbs and their complements, pointing out any faults.

The syntactic and semantic analysis are not separate as in CHAT-80 [WAR] but blended as in MICROSIAL [PIQ]. This solution is best to stop the parsing short as soon as a semantic error is detected. It is the case, for example, when a wrong complement for a verb, is encountered. In that case, ORBI always informs the user. It also provides for a more compact grammar.

full sentences

Full sentence structures are yes/no questions, wh questions and commands, with possible extrapositions.

Yes/no questions have the structure :

NP_SUBJECT VP ?

Wh questions may have the following structures :

WH_NP_COMPLEMENT NP_SUBJECT (NEGATION) VERB ?

WH_NP_COMPLEMENT (NEGATION) VERB (DOUBLE NEGATION) NP_SUBJECT ?

Or the structure :

WH_NP_SUBJECT VP ?

Commands have the structure :

(IMPERATIVE_TRANSITIVE_VERB) NP_OBJECT !

Extrapositions are acceptable in yes/no, wh questions and commands. ORBI understands left extraposition of complements of any NP_SUBJECT. The extraposed complements may occur at the beginning of the sentence and/or just before the NP subject.

Each extraposition is first syntactically analysed and concatenated to the NP_SUBJECT.

ellipsis sentences

Ellipsis may occur inside the same sentence or from sentence to sentence in the dialogue. In the first case, ORBI deals with ellipsis of subjects and/or verbs in coordinate sentencial clauses.

The interpretation of such ellipsis is not complex. The different types of gaps are expected in the grammatical rules for coordinate sentencial clauses.

Sentence to sentence ellipsis are rather more complex and differ according to the types of sentence : yes/no, wh questions and commands.

Ellipsis between yes/no questions

- ellipsis of the whole of the NP subject
- ellipsis of the whole of the VP
- ellipsis of a part of the NP subject and the whole of the VP. Examples :

Ellipsis between wh questions

- ellipsis of the whole of the WH_NP
- ellipsis of the whole of VP
- ellipsis of a part of the WH_NP and the whole of the VP.

Ellipsis between commands

- ellipsis of the verb
- ellipsis of the part of the NP object

Ellipsis occurring sentence to sentence in ORBI are solved according to the structure of the previous sentence in the dialogue. In order to do that, for each grammatical sentence accepted by ORBI one records the morphological sequences that could be ellided in the next sentence of the dialogue.

If the next sentence is not elliptical, ORBI forgets these informations. But, if ORBI tries to fill-in the gaps of the sentence in certain pre-specified ways, with the different strings recorded in an attempt to find a grammatical analysis.

The technique consists in transforming the elliptical sentence into a full (complete) sentence and to analyse it as a full sentence. The new full (ex-elliptical) sentence being morphologically built from parts of a different sentence, some syntactic controls as gender and number agreements are being suspended during the analysis. Of course, an elliptical sentence may be followed by another one.

phrase structures

Noun phrases have the following structure :

(DETERMINER) (ADJECTIVE) NOUN (COMPLEMENTS) (RELATIVE-PARTICIPLE CLAUSES)

Verb phrases have the following structures :

(NEGATION) VERB .
(NEGATION) VERB (PREPOSITION) (DOUBLE NEGATION) NP

ORBI also understands their conjunction or disjunction.

the vocabulary

ORBI's vocabulary may be divided in two parts : a core vocabulary and a specific one.

the core vocabulary (as the core grammar) is independent of the application, and transportable to other domains. It may be divided in two subparts : a metalinguistic part and a non-metalinguistic one.

The linguistic part contains : determiners, prepositions, contractions, common verbs, relative and interrogative pronouns, phrase and sentence connectors and negation, and prelocutory expressions.

The metalinguistic vocabulary contains words used to ask ORBI about its linguistic competence.

the specific vocabulary, especially nouns, abbreviations, names of entities verbs, adjectives and abverbs, refers to ORBI's domain of application.

THE SEMANTIC ANALYSIS

For each grammatical sentence a logical expression is produced, ready for evaluation, which describes its semantics, plus a term expressing the focus of attention used for output. The semantics of a sentence is obtained from that of the subject noun phrase (SNP) and from that of the quantifier freed verb phrase (VP). If the SNP is universally quantified, the final evaluation form of the logical expression differs according to whether the sentence is a yes/no one or not. Noun phrases may feature embeded quantifiers, whose scope is reversed relative to order of appearance. 'Same', 'average' and 'how many' pose special scoping problems. There is no treatment of presuppositions (the definite and indefinite articles are similarly treated), since they are unnecessary in the ORBI domain. Existential quantification only needs to be made explicit in noun phrases with 'same'.

Care is taken to produce the right order of evaluation when relative clauses (R) are present. When the relative refers to a noun (N) although it appears after its complements (C), the order (N, R, C) is used in the logical expression matrix of the noun phrase. If the relative refers to a complement, it appears immediatly after it : (N, C, R). Any universal quantification in relative clauses is always reduced to the evaluation form not(_ , not _) before their semantic embedding in the noun phrase takes place.

The semantics of verb phrases is obtained from the verb semantics V (with possible double negation) plus that of the possible ensuing noun phrase NP, where any eventual universal quantification in NP is first reduced to the form not(_ , not _).

The dictionary does not contain the semantics of individual words : they

are included in two tables. One corresponds to noun/complement pairs. Given their morphologies, their individual predicates (with appropriately linked variables) can be obtained, plus an extra predicate condition if necessary. Individual semantics of nouns can be obtained from the table as well when they occur without complements, by simply ignoring the irrelevant arguments.

A table similar to this one is used for verbs and their complements.

metalinguistic competence

Due to the wealth and variety of structures, vocabularies and uses involved in natural language, an interface for it will always fails to understand a part of the sentences produced by a user, casual or otherwise. Therefore the interface must be really informative showing explicitly why it rejects a sentence, and what its capabilities are.

As in MICROSIAL [PIQ], ORBI points out unknown words, grammatical disagreements, erroneous designations of complex entities, and false presuppositions; but also incorrect complementations of nouns and incompatible subjects and complements of verbs.

In fact all these types of diagnosis are not sufficient to ensure a good natural language interaction. In some systems, the user endeavours to consult a reference manual, generally not very successful because overly complex.

A better solution is to offer the user the possibility of asking the system questions about its linguistic competence, as we have done, to some extent, in our expert system.

The knowledge necessary to answer metalinguistic questions is grouped in a specific database.

4. Conclusions

We have found Prolog an excellent language for expert system implementation (see also [CLA]). Several sets of reasons can be adduced :

Set of reasons 1 -- It encompasses,in a single language, all desirable features of ES, by providing :

- deduction and logic context
- declarative reading suitable for knowledge representation
- modularity
- relational database
- query language
- grammar rules
- easily expressable natural language semantics
- two-way pattern matching (unification)
- non-determinism through automatic backtracking (which may be intelligent [BRU])
- self-reference (procedures can be seen as data) thus

allowing specialized control (by means of interpreters in Prolog) for query planning, explanation generation, and metaknowledge.

Set of reasons 2 -- Fast system development:

- clarity due to closeness to logical thinking, with less resort to and cleaner side-effects
- conciseness and expressiveness
- incremental modular development
- easy and improved debugging [SHA]
- built-in search strategy efficiently implemented.
- standard sophisticated data structures
- easy to write specialized interpreters on top of the language (the language is its own metalanguage)
- multiple use of procedures
- faster learning of essentials

Set of reasons 3 -- Efficient and compact programs. Prolog is susceptible of :

- Fast and concise compiled code
- Improved tail recursion optimization over functional languages
- Improved garbage collection
- Runs on small machines
- Small micro-programmable set of instructions of abstract machine
- Multiple opportunities for parallelism (unification, OR, AND)
- Intelligent backtracking

5. Acknowledgements

We are grateful to the Junta Nacional de Investigacao Cientifica for their financial support in addition to our contract with the Servico de Estudos do Ambiente of the Secretaria de Estado do Ambiente e Qualidade de Vida.

6. References

[BRU] Bruynooghe,M; Pereira,L.M.
Deduction revision through intelligent backtracking
in J. Campbell (ed.), "Issues in Prolog implementation",
Ellis Horwood, England (forthcoming), 1983.

[CLA] Clark,K ; Mc Cabe,F.
Prolog for expert systems
in Machine Intelligence n.10, Ellis Horwood, 1982.

[CLO] Clocksin,W. ; Mellish,C., Programming in Prolog, Springer-Verlag, 1981.

[OLI] Oliveira,E.,
Orbi-- an Expert Sytem as a Logic Database
Proceedings of the Conference on "Theory and Practice of Knowledge Based Systems"
Brunel University, England 1982.

[PE1] Pereira,L.M. ; Oliveira,E. ; Sabatier, P.

ORBI— an expert system for environmental resource evaluation through natural language
1st International Conference on Logic Programming, Marseille 1982.

[PE2] Pereira,L.M. ; Porto,A.
A Prolog implementation of a large system on a small machine
1st International Conference on Logic Programming,
Marseille, 1982.

[PE3] Pereira,L.M. ; Porto,A.
All solutions
Logic Programming Newsletter n.2, Autumn 1981.

[PER] Pereira,F. ; Warren,D.H.D.
Definite clause grammars for language analysis - a survey of the formalism and its comparison with augmented transition networks
Artificial Intelligence 13, 1980, 231-278.

[PIQ] Pique,J.F. ; Sabatier,P.
An informative, adaptable and efficient natural language consultable database system
Proceedings of ECAI-82, Orsay,France 1982.

[SHA] Shapiro,E.
Algorithmic program debugging
MIT Press 1983.

[WAR] Warren,D. ; Pereira,F.
An efficient easily adaptable system for interpreting natural language queries
Dept. of Artificial Intelligence, University of Edinburgh, 1981.