A PROGRAM CONVERSING IN PORTUGUESE

PROVIDING A LIBRARY SERVICE

H. M. F. Coelho

Ph.D. Thesis

University of Edinburgh

1979

ABSTRACT

TUGA is a program which converses in Portuguese to provide a library service covering the field of Artificial Intelligence.

The objective of designing the program TUGA was the development of a feasible method for consulting and creating data bases in natural Portuguese.

The resulting program allows dialogues where the program and its users behave in the way humans normally do in a dialogue setting. The program can answer, and question in pre-defined scenarios. Users can question, answer and issue commands in a natural and convenient way, without bothering excessively with the form of the dialogues and sentences.

The original contributions of this work are: the treatment of dialogues, the adaptation of Colmerauer's natural language framework to Portuguese, the particular method for evaluating the logical structures involved in Colmerauer's framework, and the library service application itself.

The program is implemented in Prolog, a simple and surprisingly powerful programming language essentially identical in syntax and semantics to a subset of predicate calculus in clausal form.

"The world of thoughts
has a model
in the world of sentences,
expressions, words, signs".

G. Frege

I declare that this thesis has been composed by myself. The work done on the development of program TUGA, while based upon Colmerauer's framework and Dahl's program, is my own and makes a number of original contributions, as discussed in Chapter 3.

Signed,

TABLE OF CONTENTS

PREFACE

This thesis describes a program, called TUGA which converses in Portuguese to provide a library service. It applies techniques developed in Artificial Intelligence (AI) and Linguistics.

The objective of TUGA is to provide users with information on an AI library. The program acts as a librarian by guiding the user in the classification of new documents and in the creation of new categories in the standard classification. The program acts as a library's secretary by giving details about the document collection and the classification system, and by adding or deleting new documents or categories.

A guide to the text follows so that the reader may select the essence of the work and skip the accessory material.

Chapter 1, "Introduction", provides a brief overview, covering all the important aspects of the work, including the nature of TUGA, research objectives, history of the work and original contributions.

Chapter 2, "The user's view of TUGA", gives a survey of what TUGA does for the user. It concludes with a description of the method of classifying documents.

Chapter 3, "Design--the main ideas embodied in TUGA", discusses the main issues which were involved in the development of TUGA, such as the way dialogue is analysed, Colmerauer's framework, definite clause grammars and the algorithm used to evaluate logical structures. The chapter ends with a brief discussion of the logical system which is used to represent Portuguese.

Chapter 4, "Implementation", describes how TUGA is organized into modules, how it works and how it makes decisions. The main procedures are described, with commentary on some representative clauses.

Chapter 5, "Related work", sets TUGA in the context of similar research carried out in AI.

Chapter 6, "Suggestions for further research", discusses the present limitations of TUGA and introduces several plans for future extensions.

Chapter 7, "Conclusion", summarizes the program's accomplishments and its social implications.

Appendix 1, "The logical system for representing Portuguese", presents the syntax and semantics of the logical system behind Colmerauer's framework, and the

definitions of some Portuguese articles.

Appendix 2, "A brief survey of PROLOG", is intended to make this text self-contained, by providing the reader with a resume' of the programming language used to implement TUGA.

Appendix 3, "Listing of TUGA", is a complete listing of the program.

Appendix 4, "Examples of dialogues", presents several examples of using TUGA.

Appendix 5, "Glossary of abbreviations used in program identifiers", provides their meaning, and it helps the reading of the program.

CHAPTER 1

INTRODUCTION

TUGA is a program which converses in Portuguese to provide a library service for the field of Artificial Intelligence (AI).

TUGA allows AI workers, even naive computer users, to ask questions, classify documents, and make bibliographies based on an AI system of categories and a document collection. In addition, the user can make changes in the document collection, and also modify the classification by inserting new ground categories. In certain situations, the program takes the initiative by posing questions to the user. It lets the user's answers constrain the program's subsequent response by making proposals about the appropriate classification of a document. It gathers document specifications, it diagnoses incomprehensible sentences, it expands the internal dictionary of Portuguese, and it creates the stock of knowledge shared by the program and its users. All of this is done in straightforward natural Portuguese.

TUGA relieves the user of the burden of learning a programming language or query language, or the formats of the data base. By having two levels of grammar, one for individual Portuguese sentences and the other for dialogues, the program knows various natural ways that a library's user may refer to the particular objects of the library world. Also, it knows the specification of each document, and the system of classification. And, it can translate a user query into a logical structure which is interpreted in terms of the contents of the data base. An appropriate answer is thereby generated according to the query form and to the retrieved data items.

NL sentence ----> logical structure ----> answer

+

data base

The processing of Portuguese queries is done in two successive stages:

i) <u>translation</u>: each user's query is transformed into a list of words and punctuation marks; syntactic plus semantic analysis is performed over the list in order to translate it into a logical structure; a dictionary containing approximately 400 words supports that analysis; the dictionary consists of a selection of general Portuguese vocabulary and AI vocabulary of single and compound

English terms; and

ii) evaluation: the logical structure is evaluated over the knowledge base; an answer or a question is generated according to the truth value found and the data retrieved; the knowledge base contains the document collection, the set of current relations and properties giving access to the collection, the classification system and the classification method.

The motivation for constructing TUGA came from the difficulty of solving rapidly the two principal problems in any library service: finding an old document and classifying a new document. In TUCA, a document is classified according to an evolving system of categories, on the basis of information supplied by the user together with information explicit in the document--namely, the references in the document text to other documents.

This task is a worthwhile subject for research because a library service is widely needed, because the library problem has a good and automatic solution, and because the mode of communication is natural. A computer user wants to have access to data dispersed through manuals, like any library user--the AI library is just an example. TUGA performs functions normally accepted as being carried out by human beings. These functions display routine performances and cognitive abilities, allowing users to explore easily the amount of data.

Users express their requests in a natural way, by conversing in Portuguese (Coelho,1979b).

The history of this work started in 1977 at Marseille, after discussions with Colmerauer and Dahl. Colmerauer had defined an interesting natural language subset to be used as an instrument to create and consult data bases (Colmerauer,1977). His framework is a general approach for translating a natural language sentence into a logical structure. Dahl had implemented Colmerauer's framework by writing a question answering program, consisting of two parts. The first part, the translation stage, was implemented as a definite clause grammar. The second part was responsible for generating answers by evaluating logical structures against a data base (Dahl,1977). The overall communication, between the program and its users, involved two sorts of languages: the user's language which supported the exchanges with the program, and the programming language which communicated the program to the computer. Dahl's program was written in Prolog, a programming language based on predicate logic and developed at Marseille (Roussel,1975).

Natural Language --> Colmerauer's framework --> Logic --> Prolog

Our work, while based upon Colmerauer's framework and Dahl's program, makes a number of original contributions, including:

-- the treatment of dialogues, ie., natural language above the sentence level;

-- the treatment of Portuguese by adapting Colmerauer's framework;

-- the evaluation of logical structures by a particular method;

-- the library service application and

-- the method for classifying documents.

The Colmerauer framework has hitherto only been applied to single and isolated sentences. In TUGA, the natural language communication is characterized by a mixed initiative and contexts. Models of dialogue and the kind of typical exchanges in a library world are integrated into a grammar of dialogues.

Colmerauer's framework is adapted to an environment of a real and useful data base of medium size where the user's language is Portuguese. The logical system for representing Portuguese has been tested over three conversational worlds -- personnel identification (Coelho,1977), civil engineering legislation (Cotta&Silva,1978), and in the library service itself.

Besides a comprehensive and rigorous definite clause grammar of the fragment of Portuguese involved, a deductive retrieval system, and a virtual relational data

base are embodied in TUGA. A particular method for evaluating logical structures is proposed. The method is suited for relational data bases.

The library service is for the first time considered in AI, as regards the construction of a program running natural language dialogues. The service is supported by a method for classifying documents. It consists of getting a list of categories for a maximum of three references, quoted in the bibliography of the document to be classified. The user discusses the content of that list, and may propose alterations to it.

TUGA is written in Prolog (Warren,1977a) running on DEC-10 systems at the University of Edinburgh, Scotland, and at LNEC in Lisbon, Portugal.

CHAPTER 2

THE USER'S VIEW OF TUGA

2.1. FORMULATION OF THE PROBLEM DOMAIN

The problems covered by TUGA belong in general to the library world, and in particular to a certain library service.

The problems are solved through dialogues between the program TUGA and its users. These dialogues occur in a precise and closed environment. It is characterized by the given knowledge sources and the kind of services delivered, by the kind of problems and questions arising currently in a library world, and by our motivation to develop a facility to improve the access to the literature needed during research activity. The library covers a restricted field -- Artificial Intelligence, but could easily be extended to other fields.

2.1.1. THE LIBRARY WORLD

The library world for TUGA is defined by;

i) a body of pre-defined information--the given data; and

ii) the kinds of questions which may be asked--possible user goals.

The given data constitute the source material to support a certain class of dialogues; and it consists of two knowledge sources:

(Gi1) - The collection of documents in Artificial Intelligence

Each document (book or paper) is described by the following attributes: author, collection number, title, publisher, date of publication, classification categories and bibliography (ie. references in the document to other literature in the same domain); and by whether it is a book or a paper.

(Gi2) - The classification system for Artificial Intelligence

The classification system is a tree of categories, each one defined by its location in the tree.

The definition of the functions of the program depend on the kind of question which may be asked in a library. The function of TUGA is deliver, receive and gather information. The possible user goals are:

(Go1) - to get information about the document collection and the classification system, either by inspecting each informational unit or by searching for related units (information receiving)

(Go2) - to add (or delete) a document from the collection (information gathering)

(Go3) - to add (or delete) a new category from the classification system (information gathering)

(Go4) - to classify an unknown document (information delivery)

2.1.2. THE LIBRARY SERVICE

Program TUGA provides a specialized service by making information available to an AI community. The main purpose of the program is informing and providing answers to questions, and encouraging the use of those materials. By encouraging the users we mean all the motivation to explore a certain collection in an easy, fast and gratifying way.

The library service is user-oriented and is defined by a user-program relationship. The user must make known his needs for information in such a way that those needs can be fully met. He must know not only how to ask questions before answers can be given, but also how to interrogate the program about its knowledge in order to achieve his goals.

The program behaves either as a librarian or as a library's secretary. The librarian is an information specialist able to assist the user while he is asking questions about documents. The librarian presents the user with a classification system and a classification method which are used either to provide information or to guide an exploratory search of the library. The library's secretary is a "go-between" who only provides a user with information (question with answer), either explicit or implicit in the document collection.

2.2. THE DOCUMENT COLLECTION

The prototype document collection has 46 items (21 books and 25 papers), and includes the AI publications which were most frequently referenced in IJCAI-77 (cf. Sigart Newsletter no.65, 1978), and also well known works. Each document is described by the following attributes:

whether it is a book or paper;
author;
number;
title;
publisher;
year;
classification categories;
bibliography.

2.3. THE CLASSIFICATION SYSTEM

The classification system is an open-ended scheme of 42 categories hierarchically arranged, where each category may be subordinate to at least one broader category. New categories may be inserted by user initiative to improve the system.

The classification system mainly covers the field of AI, the subject of the document collection, but viewing AI as a part of a broader field, the Computing Sciences. The system also shows their connection with related disciplines, such as Linguistics and Logic. The classes of system categories embrace AI basic methodologies and techniques and their current major application areas (Nilsson,1974). Their organization follows the ACM proposal for the categories of the Computing Sciences. The classification system is presented in the figure below.

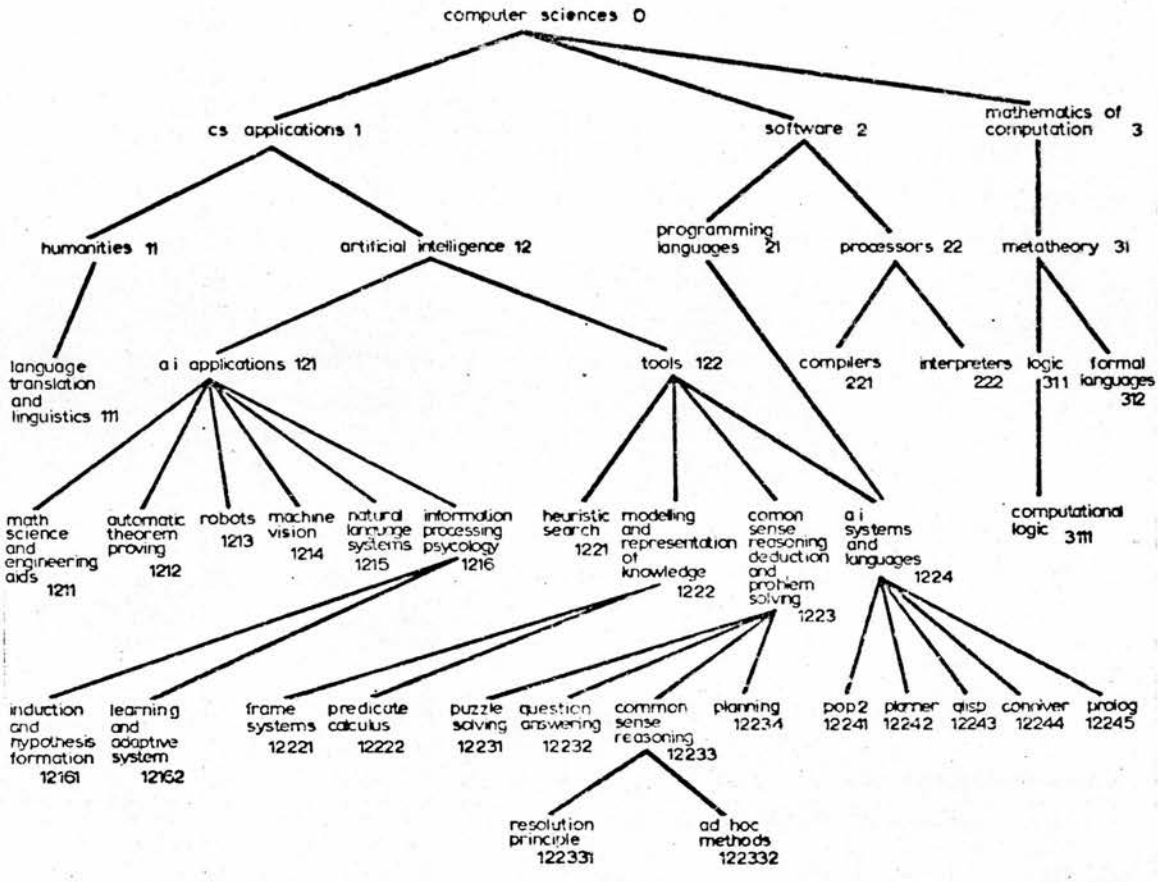


Fig.1

Classification system for Artificial Intelligence

2.4. THE CLASSIFICATION METHOD

The classification method we have adopted is based upon the classification assigned to the bibliography referenced in a document. The program asks for a maximum of three references and proposes a set of selected categories as the classification for that document. The user may accept it; or change it through deletion of any category and insertion of more appropriate ones.

This method is based upon the interaction of two points of view: i) that of the program, which knows the classification ascribed to the referenced bibliography of the document collection stored in the data base, and ii) that of the user, who knows the content of the document to be classified, some of its references and/or the classification system. The classification of a document is considered as a problem whose solution involves the cooperation of program and user, through a conversational process. So, the problem-solving ability is divided between the two participants. The program's ability is essentially directed to attaining a first solution for the problem statement, to discussing it with the user, and to checking the reliability of a second solution to be proposed by the user.

In brief, the classification method consists of the following steps:

-- The program asks for a maximum of three known references of the document to be classified; the user may know only one or none!

-- The program proposes to the user a set of categories for the classification of the document; the final set is obtained by eliminating the redundant ones: the repeated ones and the ones placed above some listed category. When the program does not know any of the references given by the user, it asserts its inability to classify the document, and accepts any classification proposed by the user.

-- The user may ask the program about its knowledge during the classification process.

-- The user may accept the proposal or modify it by suggesting at least one different category; the program inspects the new set and rejects any unknown category.

The following two dialogue segments, between the program (p) and one of its users (u), exemplify some aspects of this classification method.

Dialogue 1

.
.
.

u-Classifique "Experimental learning model"! (Classify "Experimental learning model"!)

p-"experimental learning model" e´ desconhecido! Quer ainda classifica´-lo? ("experimental learning model" is unknown! Do you still want to classify it?)

u-Ok.

p-Vamos classifica´-lo! Por favor, de-me o titulo de uma referencia do documento em questao. (Let us classify it! Please give me the title of one reference from the document in question.)

u-"Artificial intelligence".

p-"artificial intelligence" e´ conhecido, e esta´ classificado: nas categorias: 12--"artificial intelligence". Por favor de-me o titulo de outra referencia do documento em questao. ("artificial intelligence" is known, and it is classified under the categories:... Please, give me the title of another reference from the document in question.)

u-Nao tenho mais nenhuma. (I have no more.)

p-Esta´ bem! Vamos utilizar o metodo de classificacao que consiste em sugerir como categorias para o documento em questao, (All right! Let us adopt as a classification method one that consists of suggesting as categories

as obtidas da referencia fornecida pelo utilizador! Assim, proponho que o documento fique classificado nas categorias seguintes:
12.
Por favor faca a sua escolha tendo em atencao as categorias sugeridas e as que julgue mais apropriadas
De-me 3 categorias no maximo!

of the document in question, the ones obtained from the reference given by the user! Thus, I propose that the document be classified under the following categories:12.
Please choose according to the suggested categories and the ones that you think most appropriate.
Give me a maximum of three!)

u-12, 111 e 1224.

p-O documento ficou classificado nas categorias:
12--"artificial intelligence",
111--"language translation and linguistics" e
1224--"artificial intelligence systems and languages".

(The document was classified under the following categories:

Dialogue 2

u-Classifique "A structure for plans and behaviour"!

(Classify "A structure for plans and behaviour"!)

p-"a structure for plans and behaviour" e´ desconhecido!
Quer ainda classifica´-lo?

("a structure for plans and behaviour" is unknown!
Do you still want to classify it?)

u-Sim, quero.

(Yes, I do.)

p-Vamos classifica´-lo!
Por favor de-me o titulo de uma referencia do documento em questao.

(Let us classify it!
Please give me the title of one reference from the document in question.)

u-"Computational semantics".

p-"computational semantics" e´ conhecido, e esta´ classificado nas categorias!
1215--"natural language

("computational semantics" is known, and is classified under the following categories:
.....

```
        systems" e
  312--"formal languages".
  Por favor, de-me o titulo   Please give me the title of
  de outra referencia do       another reference from the
  documento em questao.            document in question.)

u-"Computer models of
  thought and language".

p-"computer models of                ("computer models of
  thought and language" e'         thought and language" is
  conhecido, e esta'               known, and is classified
  classificado nas categorias:       under the categories:
  1215--"natural language                          .....
  systems" e
  1222--"modelling and
  representation of knowledge".
  Por favor, de-me o titulo   Please give me the title of
  de outra referencia do       another reference from the
  documento em questao.            document in question.)

u-"Computer power and human reason".

p-"computer power and human          ("computer power and
  reason" e' conhecido, e        human reason" is known, and
  esta' classificado na              is classified under the
  categoria:                                      category:
  1215--"natural language                          .....
  systems".
  Vamos utilizar o metodo de                Let us adopt as
  classificacao que consiste     classification method the
  em sugerir como categorias                   one that consists
                                 of suggesting as categories
  para o documento em           of the document in question,
  questao, as obtidas das 3      the ones obtained from the
  referencias fornecidas          three references given by
  pelo utilizador!                the user!
  Assim, proponho que o       I therefore propose that the
  documento fique classificado      document be classified
  nas categorias seguintes:           under the following
  1215, 312 e 1222.                       categories:...
  Por favor, faca a sua       Please choose according the
  escolha tendo em atencao        suggested categories and
  as categorias sugeridas     the ones that you think most
  e as que julgar mais apropriadas.          appropriate.
  De-me 3 categorias no                 Give me a maximum
  maximo!                               of three!)

u-1222.

p-O documento ficou classificado   (The document has been
  na categoria:              classified under the category:
  1222--"modelling representation                  .....
  of knowledge".
```

Por favor, deseja arquivar este documento na Base de Dados?

Please do you want to store this document in the Data Base?)

u-Ok.

p-Por favor, qual o nome do autor do documento?

(Please what is the name of the author of the document?)

u-Sacerdoti.

p-Por favor, qual o nome do livreiro do documento?

(Please what is the name of the publisher of the document?)

u-Elsevier.

p-Por favor, qual a data de publicacao do documento?

(Please what is the date of publication of the document?)

u-1977.

p-Por favor, qual e´ o tipo do documento?

(Please what is the type of the document?)

u-livro.

(book.)

p-O novo documento no.47, e a restante informacao adicional, foi arquivado! E que mais?

(The new document no.47, and the additional information, has been stored! And what else?)

.
.
.

2.5. PROGRAM REASONING

Program reasoning is handled by inference mechanisms, general or special purpose procedures, able to support its competence to converse.

-- General problem reasoning

As examples of general purpose mechanisms we have those dealing with:

(1) remembering and numbering the participant's contributions

Each contribution (eg. a user answer) is numbered and assigned to the corresponding participant, allowing the building of the dialogue history. Thus it is possible to refer to the present state of the dialogue and to use the previous contribution to take decisions. This facility makes it easier for the program to understand or participate in nested dialogues.

(2) Answering constraint

Answering involves the output of data. Sometimes the amount of data is very large, and it may be unexpected for the user. In such case, it is convenient to output only the first items, and then inform the user that there are more data to be supplied.

The following dialogue samples illustrate this kind of mechanism (the token '=>' indicates the program's questions):

```
:
:
:

u-Quais as categorias                (What are the categories
  acima de Prolog?                          above Prolog?)
```

```
p-As cinco primeiras sao:                    (The first five are:
  computer sciences,                          .....
  computer sciences applications;             .....
  software;                                   .....
  artificial intelligence e                   .....
  programming languages.                      .....
  Existem mais 2!                            There are two more!
=>Quer mais?                                  Do you want more?)

u-Sim.                                                    (Yes.)

p-As restantes sao:                          (The remainder are:
  artificial intelligence tools e           .....
  artificial intelligence systems and languages      ....)

  .
  .
  .

u-Qual e' a bibliografia para "Artificial Intelligence"?
  (What   is   the   bibliography   for   "Artificial
  Intelligence"?)

p-Os primeiros cinco documentos sao:
  (The first five documents are)

       livro  no.   2  --  "problem-solving  methods  in
                           artificial intelligence";

       livro no.  6 -- "a computational  model  of  skill
                           and acquisition";

       livro no.  7 -- "human problem solving";

      livro no.  10 -- "computer power and human reason"
                           e

      livro no.  23 -- "STRIPS:  a new  approach  to  the
                           application  of  theorem proving to
                           problem-solving".

  Existem mais 3 documentos!          (There are three more
                                                documents!)

=>Quer mais documentos?               (Do you want any more
                                                documents?)

u-Quero mais dois documentos.         (Yes,I want two more
                                                documents.)

p-Os restantes sao:                          (The remainder are:
  artigo no.  25 -- "the CONNIVER reference manual" e
  artigo no.  27 -- "planning in a hierarchy of
                     abstraction spaces".
```

=>E que mais? (And what else?)

The program knows that the output of data objects is constrained by their number. The program outputs the first five objects; states the number of missing ones and interrogates the user about their output.

(3) Dialogue shifting

During a dialogue on a precise topic, the user may give up, and restart a new dialogue.

```
    .
    .
    .
u-Classifique "Explorations in cognition"!   (Classify "Explorations in cognition"!)

p-"explorations in cognition" e´ desconhecido!   ("explorations in cognition" is unknown!
=>Quer ainda classifica´-lo?   Do you still want to classify it?)

u-Por favor, quero.   (Yes, please.)

p-Vamos classifica´-lo!   (Let us classify it!
=>Por favor, de-me o titulo de uma referencia do documento em questao.   Please, give me the title of a reference from the document in question.)

u-Desisto.   (I give up.)

p-Ok. Vamos desviar-nos desta conversa!   (Ok. Let us change conversation!
=>E que mais?   And what else?)
    .
    .
    .
```

The program recognizes whether the document to be classified is known. In the case of a new document, it interrogates the user to confirm his request, and starts its classification by asking pre-defined questions, such as that on the title of a reference.

(4) gathering missing information about the document

After the user has decided to store a newly classified document, the program is able to gather the missing relevant document specification. The program knows what to ask, and is able to respond to the user. The program is also able to handle user changes of mind, contradictions and repetitions occurring during the corresponding dialogue. The following dialogue sample illustrate some aspects of this kind of mechanism;

.
.
.

p-Por favor, deseja arquivar este documento na Base de Dados? (Please, do you want to store this document in the Data Base?)

u-Por favor. (Please.)

p-Por favor, qual o nome do autor do documento? (Please, what is the name of the author of the document?)

u-Brazdil e 1978.

p-Por favor, qual o tipo do documento? (Please, what is the kind of document?)

u-artigo. (paper.)

p-O novo documento no. 50 com a restante informacao (The new document no. 50 with the other

adicional foi arquivado! information has been stored!
E que mais? And what else?)
.
.
.

Program question on the year of the document was unnecessary because the user gave that information with the name of the author.

(5) handling the classification method

During the process of classifying a document the program is able to simplify the set of categories of its references. It verifies any repeated ones and it consults the classification system to find redundant categories, ie. those situated above other categories belonging to that set. A previous example of a conversation, presented to illustrate the classification method, shows this kind of inference mechanism.

(6) diagnosis of incomprehensible sentences

During the dialogue the program may not understand the user's sentence. This may be due to the following:

-- the user has failed to type a word properly;
-- the user has employed a word unknown to the program, and
-- the user has adopted a syntactical construction unknown to the program.

For the two first cases, the program discovers the unknown word(s), asks the user about it(them), and is able to increase its dictionary with the new word if it is a proper noun. In the third case, it asks the user to paraphrase his sentence.

The following dialogue sample illustrates the second case:

```
   .
   .
   .
u-Michie e´ um autor?                    (Is Michie an author?)

p-Nao compreendo esta frase          (I don´t understand this
  porque a palavra "michie" e´         sentence because the
  desconhecida!                    word "michie" is unknown!
  Por favor, preste atencao `a         Please, observe with
  frase que escreveu! care the sentence you have typed!
=>Existe algum erro sintatico                  Is there any
  na escrita da sua frase?           syntactical mistake?)

u-Nao existe.                                (There is none.)

p-Esta´ bem!                                       (All right!
=>A palavra desconhecida "michie"    Is the unknown word
  e´ um nome proprio?                "michie" a proper noun?)

u-E´.                                              (It is.)

p-Qual e´ o genero de "michie"?                  (What is the
=>                                         gender of "michie"?

u-Masculino                                    (Masculine.)

p-A qual dos tipos, autor,           (To which type, author,
=>titulo, livreiro ou      title, publisher or category,
  categoria pertence a palavra                 does the word
  "michie"?                                  "michie" belong?)

u-Autor.                                          (Author.)

p-A respostaa sua pergunta                 (The answer to your
  sobre "michie" e´:               question about "michie" is:
  Nao.                                              No.)
   .                                    (ie. no known document
   .                              has "michie" as its author.)
   .
```

-- Linguistic program reasoning

A typical example of linguistic program reasoning is the following dialogue consisting of an anaphoric question with the program's answer:

u-Quem e' o autor do artigo?
(Who is the author of the paper?)

p-A sua pergunta e' indefinida!
Nao encontro nenhuma entidade que a satisfaca.
(Your question is undefined! I don't find any appropriate entity.)

This dialogue illustrates another feature of the program. TUGA evaluates the logical structure and finds that the truth value is undefined. The question reveals a definite noun phrase anaphora. The noun phrase 'o artigo' (the paper) is meaningless in a context where no paper has been suggested previously: the question is isolated. Also there are no co-reference sentential restrictions and the noun phrase cannot be interpreted as referring to a certain paper. The presupposition introduced by the definite article is solved but not found.

CHAPTER 3

DESIGN--THE MAIN IDEAS EMBODIED IN TUGA

3.1. ANALYSIS OF DIALOGUE

The organization of taking turns to talk is fundamental to dialogue, as well as to a program able to converse. A formalism for the analysis of dialogue is proposed, and we examine its compatibility, with the representation of the history of dialogues between a program and its users. Dialogue taking place in a library world has structural properties, and rules are derived to form a grammar of dialogues. This grammar is responsible for the organization of the possible interactions in such a context. However, some of these rules are general and may apply to other contexts. For example, rules for opening dialogues and for allowing user initiative are general. Rules are defined in terms of semantic concepts, like requests or answers, which are supported by sentences of natural language and analysed according to Colmerauer's framework.

-- A formalism for the analysis of dialogues

Let P be a set of participants and C a set of contributions. By a contribution act we mean a member of the set $P \times C$ of participant-contribution pairs. For example,

$$\langle p1, c11 \rangle$$

is a contribution act, where p1 and c11 are the first members of P and C, respectively.

Let S be a set of conversational states or configurations. A conversational state s is a sequence of at least two related contribution acts. For example,

c11

stands for the first contribution regarding the first conversational state.

By a dialogue of length n we mean a member of the set $(P \times C)^n$ of sequences of n contribution acts; and by a dialogue we mean a member of the set

$$T = \bigcup_n (P \times C)^n \qquad (n \in N)$$

of dialogues of any length. Each member of a dialogue is of the form

$$\langle s, \langle p, c \rangle \rangle \qquad (s \in S, p \in P, c \in C)$$

which we identify with the triple

$$\langle p, s, c \rangle$$

For example,

$$T=\{\langle p1,1,c11\rangle,\langle p2,1,c21\rangle,\langle p1,2,c32\rangle,\langle p2,2,c42\rangle\}$$

is a dialogue of length 4, with 2 participants, p1 and p2, 2 conversational states and 4 contributions.

We call $E=P\times S\times C$ the set of events, and any triple $\langle p,s,c\rangle$ an event.

A dialogue is a sequence of events, grouped into units, and governed by rules.

The conversational units are the invariant structures of dialogue: sub-dialogues, exchanges, monologues and contributions.

A sub-dialogue (dialogue course or segment) is any sequential subset of a dialogue,

$$Course=\{x:(\exists\, t\in T)(x\cap t=0 \text{ and } x\cup t\in T)\}$$

An exchange is a set of two consecutive events, concerning the same conversational state and two different participants. A pre-defined exchange, conducted by the program, is called an exchange pattern.

A monologue is a sequence of at least two consecutive events, concerning the same conversational state and the same participant.

$$Mon_p=\bigcup_n(\{p\}\times C)^n$$

A contribution of a participant to a dialogue is a sequence of his contributions.

The semantics of contributions covers the following types: requests (statements, questions, and commands), answers and remarks (eg. agreement).

The underlying structures of the situations occurring in a certain problem world determine the organization of dialogue and its systems.

A grammar of dialogues is a set of rules of dialogue.

Rules of dialogue state how participants understand coherent dialogues, and specify the membership of the set of legal dialogues K, such that

$$K \subset T$$

where T is the set of dialogues of any length.

Rules of dialogue define the class of coherent dialogues and their attached models. They contain the way contributions are put together.

A model is a system of dialogue defined as the triple

$$\langle P, C, K \rangle.$$

The core of any model is the set R of rules defining K.

-- A particular class of dialogues

We consider the dialogues occurring between the program TUGA and its users in the library world. Such dialogues are evolving dual processes, goal-and-rule-oriented for sharing information between the participants. They are dual because there are only two participants at a given time. They are goal oriented because they are carried on to satisfy, for example, the following objectives:

i) to satisfy users' straightforward requests concerning the document collection and the classification system;

ii) to ask users about the library world (eg. the author of a paper), for conversational guidance purpose, and

iii) to present the user with proposed data (eg. the document classification), enabling him to choose from or modify it.

They are rule-oriented because the conversational units are governed by a grammar of dialogues which determines the roles played by the program--the librarian and the library's secretary, and by the user. The grammar also allows the inclusion of several courses inside a dialogue, such as topic shifts for simple question-answering purposes.

Consider a dialogue between the program and one of its users, defined as a sequential organization of single exchanges. The conversational description is represented in the diagram of the following figure.

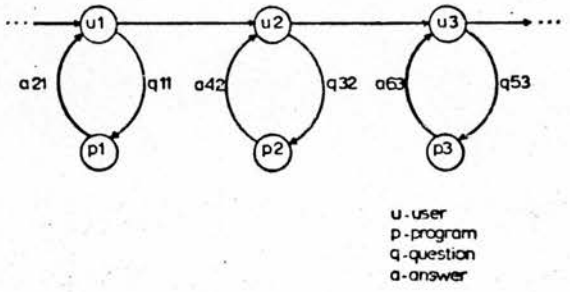


Fig.2

Conversational description

The diagram shows the history of that dialogue. The interaction proceeds in separate discrete steps from one to another of a finite number of configurations. There are three successive conversational states through time, each one corresponding to a single exchange, question(q)-answer(a) pair, conducted under user initiative.

This history is described as follows,

T={<u,1,q11>,<p,1,a21>,<u,2,q32>,<p,2,a42>,<u,3,q53>,
<p,3,a63>}

ie. a dialogue of length 6, with 2 participants, the user(u) and the program(p), with 3 conversational states and 6 contributions (questions and answers). Note that 'q32', for example, means that the third contribution is

a question, referring to the second conversational state.

Consider now the following segment of a nested dialogue, concerning the classification of a document:

.
.
.

(1) u-Classifique "Explorations in Cognition"! (Classify "Explorations in cognition"!)

(2) p-"explorations in cognition" e´ desconhecido! ("explorations in cognition" is unknown!
(3) Quer ainda classifica´-lo? Do you still want to classify it?)

(4) u-Esta´ bem. (All right.)

(5) p-Vamos classifica´-lo! (Let us classify it
(6) Por favor, de-me o titulo de uma referencia do documento em questao. Please, give me the title of a reference from the document in question.)

(7) u-Qual e´ o titulo do Nilsson? (What is the title of Nilsson?)

(8) p-"artificial intelligence". ("artificial intelligence".
(9) Vamos classifica´-lo! Let us classify it!
(10) Por favor, de-me o titulo de outra referencia do documento em questao. Please, give me the title of another reference from the document in question.)

(11) u-Nao tenho nenhuma. (I have none.)

(12) p-Esta´ bem! (All right!
(13) Ja´ que nao consegui obter nenhuma referencia, nao sei dar-lhe qualquer sugestao! Quer ainda classifica´-lo? As I could not get any reference, I don´t know how to give you any suggestion! Do you still want to classify it?)

(14) u-Nao quero. (No, I don´t.)

(15) p-Esta´ bem! E que mais? (All right! And what else?)

.
.
.

This dialogue presents a sub-dialogue and an interruption due to the user. The sub-dialogue corresponds to a dialogue model, which organizes interactions concerning the implementation of the document classification method. The interruption is presented as an exchange, nested in the sub-dialogue. The conversational description, represented in the figure below, shows the organization and levels of the dialogue. There are three levels: one for the ordinary dialogue; the other for the particular dialogue model; and, the last one for the interruption. Numbers, placed on the left of contributions, are used for referencing them to the segment considered.

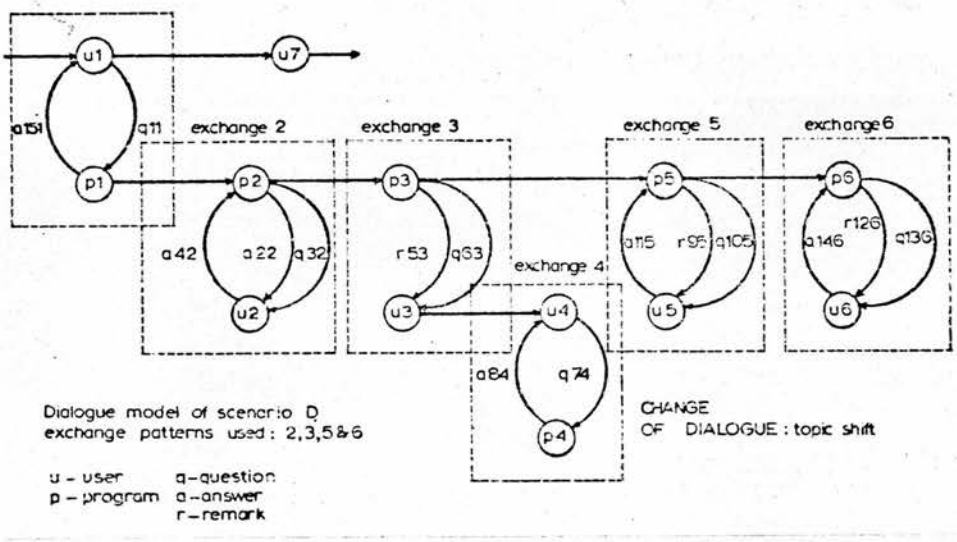


Fig.3

Conversational description

The history of this dialogue is described as follows,

```
T={<u,1,q11>,<p,2,a22>,<p,2,q32>,<u,2,a42>,
   <p,3,r53>,<p,3,q63>,<u,4,q74>,<p,4,q84>,
   <p,5,r95>,<p,5,q105>,<u,5,q115>,<p,6,r126>,
   <p,6,q136>,<u,6,a146>,<p,1,a151>}
```

ie. a dialogue of length 15, with 2 participants, the user(u) and the program(p), 6 conversational states and 15 contributions (questions(q), answers(a) and remarks(r)).

--A grammar of dialogues

The grammar of dialogues of program TUGA is a complete and precise description of the properties of a certain class of dialogues. The properties concern the structures of the dialogues, occurring in a library world, and organized as models.

A dialogue carried out by TUGA has two participants, the program (p) and its users (u), and therefore two mutually exclusive states, the "agent" and the "passive participant". Both participants may take the initiative during the encounter, ie. the program may be an "agent" or a "passive participant". The "agent" claims the turn to speak at any given moment, and plays an active role. The "passive participant" does not claim the turn to speak at any given moment.

Considering two states for each participant, there are four possible conversational states. However, we only consider two states: "agent" - "passive participant" and "passive participant" - "agent". (The other two states represent in some sense a failure of dialogue.)

The syntax above the discourse level, presented below, characterizes only the class of dialogues considered. For more details, the reader is asked to consult the program TUGA, and in particular, its module CONVER (see Appendix 3). This matter is discussed further on in Chapter 4.

Grammar of dialogues

Syntax above the discourse level

```
<converse>    -->  <opening1>, <converse>
<converse>    -->  <opening2>, <converse1>

<converse1>   -->  <converse2>, <continue>

<converse2>   -->  <dialogue>
<converse2>   -->  <monologue>


<converse3>   -->  <dialogue1>
<converse4>   -->  <decide>, <dialogue>
<converse5>   -->  <dialogue2>, <course>
<converse6>   -->  <ask_author>,
                   <ask_publisher>,
                   <ask_date_of_publication>,
                   <ask_document_type>
```

```
<converse7>  -->  <converse2>, <dialogue3>,
                  <converse2>


<continue>   -->  <conversel>
<continue>   -->  <suspend>, <converse>
<continue>   -->  <close>


<dialogue>   -->  <user>, <program>
<dialoguel>  -->  <p_questionl>, <dialogue>
<dialogue2>  -->  <p_question2>, <dialogue>
<dialogue3>  -->  <p_question3>, <dialogue>

<monologue>  -->  <user>, <dialogue>


<course>     -->   <converse4>
<course>     -->   <refusal>, <converse4>
<course>     -->   <change>
<course>     -->   <dialogue3>, <refusal>
<course>     -->   <return>, <converse3>


<user>       -->   <question>
<user>       -->   <fact>
<user>       -->   <command>
<user>       -->   <answer>


<program>    -->   <response>
<program>    -->   <response>, <converse3>
<program>    -->   <response>, <converse5>
<program>    -->   <response>, <converse6>
<program>    -->   <response>, <converse7>
```

N.B. We adopt the TUGA's predicates whenever possible.

Let us consider only the first few rules in order to make explicit their meaning. A general dialogue, 'converse', is defined as an opening followed by a

sub-dialogue which may be followed by a sub-dialogue or closed by user initiative. The user may also suspend temporarily the dialogue without affecting it. This feature justifies the existence of two kinds of opening: one for the dialogue start and the other for the re-start. A dialogue is simply a sequence of exchanges or monologue, or is followed by several models of dialogue. For example, dialogue on the classification of a document is handled by dialogue model 'converse5'; dialogue on adding new documents is handled by dialogue model 'converse6'. Dialogue model 'converse5' is defined by rules 'course' which define several kinds of possible courses during the interaction between program TUGA and its users. Any of these dialogue models is served by sets of exchange patterns. For example, 'converse6' is defined as a sequence of pre-defined program questions whose order may be altered by user. This means that the program can cope with single or multiple data, provided in any order, and can avoid asking questions whose answers were provided either implicitly or explicitly at some earlier time. This particular model is also able to deal with user changes of mind.

We use rules of interpretation, below the discourse level. The rules of interpretation deal with what the user does, eg. requests (statements, questions and commands) and answers. Other rules deal with what the program does, eg. answers, questions, and remarks

(comments and agreements). Here are, for example, three of these rules:

Rule--If the user makes a statement, the program interprets it as a request for confirmation.

Rule--If the user asks a closed question (form Q-S, where S means the statement corresponding to the question) and the program responds with an existential E (yes/no), then the program is understood as answering the user with the statement E-S.

Rule--If the user issues a command, then the program interprets it as a valid request for an action A only if the following conditions hold:

the request is ended with an exclamation mark,
and action A is one of the following

- classifying a document,
- generating a category
- adding data items and
- deleting data items.

The first two actions also cover the general purpose of gathering information through a referent: the referenced document or the classification.

3.2. THE COLMERAUER ANALYSIS OF NATURAL LANGUAGE

Colmerauer's framework (Colmerauer,1977) to natural language analysis provides us with a method for translating natural language sentences into logical structures.

The method consists of considering,

- elementary statements based on proper nouns,
- each article as a three-branched quantifier, and
- four precedence rules for governing the quantification hierarchy problem.

Example 1: the sentence,

Hewitt is (a) writer.

constructed with a noun and the verb 'to be' is translated into the formula,

writer(hewitt)

In general, verbs, adjectives and nouns introduce properties with n arguments. For verbs, n may be equal to 1 (intransitive verbs) or N+1 (transitive verbs, where N is the number of complements). For adjectives and nouns n is equal to 1 or greater than 1 (relations, where n is its n-arity). The arguments represent objects, whose role in a sentence is the complement of a noun, verb or adjective.

Example 2: the sentence,

Hewitt writes a book.

constructed with a verb ('write'), a noun ('book') and an article ('a'), may be replaced by the following paraphrase,

for a
B
such that
B is (a) book (1)
it is true that
Hewitt writes B (2)

where (1) and (2) are elementary statements.

This paraphrase is a logical structure written in a shorthand notation,

a(B,book(B),writes(hewitt,B))

and represented by the tree,

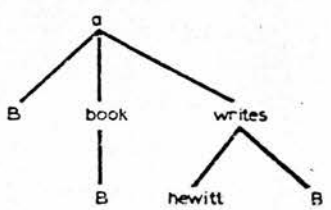


Fig.4

Tree structure of a sentence

Note that statements (1) and (2) are translated into formulae 'book(B)' and 'writes(hewitt,B)', respectively.

The logical structure is the meaning of the sentence, and each of its constituent parts corresponds to the senses of individual words (Frege's principle). We shall refer to the representations of such meaning as logical structures since the only aspects of meanings we know how to represent rigorously are logical relations.

Each article α introduces a three-branched quantifier q, which creates a new formula from a variable x and two formulae f1 and f2,

$$q(x, f1, f2)$$

corresponding to the statement,

"for α x such that e1, it is true that e2"

where e1 and e2 are the elementary statements corresponding to f1 and f2.

Example 3: the sentence,

Hewitt writes a book for each publisher.

constructed with a verb ('write'), two nouns ('book', 'publisher'), and two articles ('a', 'each'), may be replaced by the following paraphrase:

> "for each P such that P is a publisher it is true that for a B such that B is a book, it is true that Hewitt writes B for P".

The sentence is translated into the logical structure,

```
each(P,
     publisher(P),
     a(B,
       book(B),
       writes_for(hewitt,B,P)))
```

and represented by the tree:

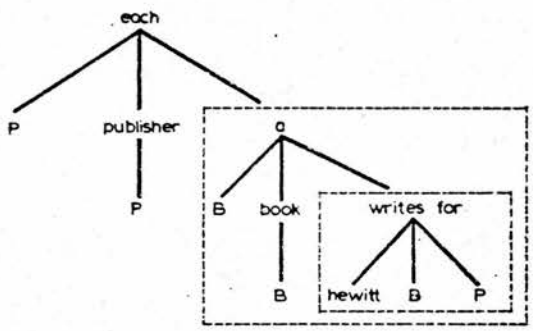


Fig.5

Tree structure

The logical structure displays the following precedence rule: "in a construction involving a noun and a complement of this noun, the quantification introduced by the article of the complement dominates the quantification introduced by the article of the noun".

Besides this rule, Colmerauer proposed three more precedence rules:

-"the quantification introduced by the article of the subject of a verb dominates the quantification(s) introduced by the complement(s) closely related to that verb", for example, the direct and indirect complements, and other complements for specific verbs;

-"whenever a verb, an adjective or a noun has two complements, the quantification is made in the inverse order of the natural order of their appearance; the rightmost complement generates a quantification dominating the quantification generated by the other complement";

-"the quantification(s) introduced by the subject dominate(s) the negation" (exception: when the subject is quantified by 'every' or 'each', negation dominates).

Complex sentences involving relative clauses of the restrictive type require a rule similar to the one necessary for the treatment of coordinated statements. Relative clauses are treated as ordinary statements: the relative pronoun is replaced by the appropriate variable and the whole is linked to the translation of the noun by the conjunction 'and'.

Example 4: the sentence,

> Winograd wrote the book that is referenced in "Psychology of Computer Vision".

is translated into the logical structure,

```
the(x,
    and(isbook(x),
        isreferenced(x,"psychology of computer vision")),
    wrote(winograd,x))
```

and is represented as the tree,

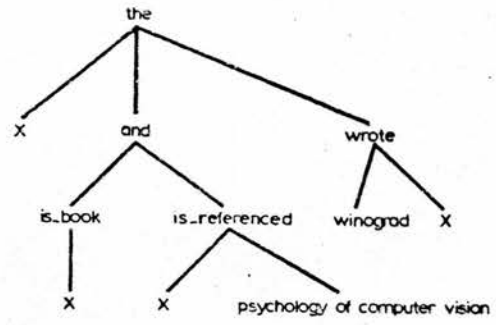


Fig.6

Tree structure

Observe that the relative pronoun was eliminated and the scope of 'and' is only a part of the sentence. The conjunction 'and' operates upon a particular book, adding the pieces of information asserted by the sentence.

3.2.1. ADAPTING THE COLMERAUER ANALYSIS TO PORTUGUESE

Colmerauer framework was originally proposed for French and English. Later on (Dahl,1977) adapted it to Spanish and (Pique,1978) suggested a different semantics for French articles.

The adaptation to Portuguese covers the definition of articles, the double negation imposed by the use of 'nenhum' (no), the treatment of the conjunction or disjunction of proper nouns, the conjunction of statements, questions, common nouns and adjectives, the use of the relative/interrogative pronoun 'quem' (who)

only with nouns that refer to humans, the particular use of special verbs and adverbial groups, and definite articles with proper nouns. Examples of sentences, occurring in the library world, are presented in section 4.4.1 to show a brief survey of Portuguese grammar.

The definitions of Portuguese articles (see Appendix 1) are slightly different from Colmerauer's and Dahl's, especially as regards the articles translatable by 'a', 'no' and 'not all'. A new article 'the i' ('i' is an integer) was introduced for the first time. The reason for these differences lies in what we consider the semantic interpretations of the Portuguese articles. Other articles like 'somebody', 'nobody', 'any' and 'each' are handled by the definitions of articles 'a', 'no' and 'all the', respectively.

The use of 'nenhum' (no) imposes a double negation in some situations, characterized by the appearance of particle 'nao' (not). This use is governed by rules based upon the location of 'nenhum' in the sentence with regard to the verb (see section 4.4.6).

The treatment of coordinated proper nouns and the disjunction of proper nouns is achieved through the introduction of compound terms, such as 'coord(x)' and 'disj(x)', in the vocabulary of the logical system (see Appendix 1), as it is explained in Chapter 4.

Care is taken in the treatment of special verbs, like 'to be'. The verb tense ellipsis for 'to be', very common in Portuguese, is allowed, and the verb 'to have' is defined carefully (see section 4.4.6), taking also into account constructions involving adverbial groups.

Proper nouns may occur either with the definite article or without it. Both uses are allowed.

3.3. EXPRESSING THE COLMERAUER ANALYSIS AS A DEFINITE CLAUSE GRAMMAR

We consider now how to express the Colmerauer analysis, the translation of sentences of natural language into logical structures, as a definite clause grammar (DCG).

DCGs (Pereira&Warren,1978), an extension of context-free grammars, support the construction of the translation machinery, which tries to capture some of the syntax and the semantics of the Portuguese subset, relevant for the application chosen. The translation consists of the assignment of a logical structure as the interpretation of every sentence. This structure is composed of well-formed formulae of a certain logical system, based upon an extension of predicate logic. The grammar machinery is expressed as a set of definite clauses of logic through Prolog grammar rules. It

contains syntactic and semantic knowledge of the subset of the natural language considered.

First, we take into account only syntax and we explain what the parsing of a sentence consists of, through its representation in predicate logic as the proof of inconsistency of a set of clauses (Kowalski,1974; Emden,1975). Second, we take into account syntax and semantics and we present a simplified grammar, able to analyse a small fragment of English. We show how it is constructed, and how it parses and translates a sentence.

Example 1: The parsing problem is formulated in the following way: "given a grammar and an initial string of words demonstrate that the string is a sentence" (Knuth,1971).

The problem reduces to the construction of a parse tree for verifying whether the parsed string is grammatical. The parse is the path in the tree of goal statements from the root (initial sentence) to the halt statement.

The specification of the parsing problem contains the transcription of a context-free grammar, CFG, and a representation of the string of words.

Here is an example of a CFG (a simplified version of TUGA's grammar), defined by ten rules (1):

```
sentence --> noun_phrase, verb, complements.
noun_phrase --> article, adjective, common_noun.
noun_phrase --> article, common_noun.
complements --> noun_phrase.

verb --> [writes].
article --> [the].
article --> [a].
adjective --> [new].
common_noun --> [author].
common_noun --> [paper].
```

Each rule has the form:

```
non_terminal_symbol --> body.
```

where body is a sequence of one or more items separated by commas. Each item is either a non-terminal symbol or a sequence of terminal symbols. The meaning of the rule is that body is a possible form for a phrase of type non-terminal symbol. A non-terminal symbol is written as a Prolog atom, while a sequence of terminals is written as a Prolog list, where a term may be any Prolog term.

Grammar CFG covers sentences such as:

"the new author writes a paper"

To get the translation of each rule of a CFG into a definite clause of logic, we associate with each non-terminal a 2-place predicate (having the same name).

---

(1)The Prolog notation is herein adopted(see Appendix 2).

The arguments of the predicate represent the beginning and end points in the string of a phrase for that non-terminal. The ten rules of that CFG are translated into:

```
(1)  sentence(S0,S):- noun_phrase(S0,S1),
                      verb(S1,S2),
                      complements(S2,S);
(2)  noun_phrase(S0,S):- article(S0,S1),
                         adjective(S1,S2),
                         common_noun(S2,S);
(3)  noun_phrase(S0,S):- article(S0,S1),
                         common_noun(S1,S);
(4)  complements(S0,S):- noun_phrase(S0,S);

(5)  verb(S0,S):- connects(S0,writes,S);
(6)  article(S0,S):- connects(S0,the,S);
(7)  article(S0,S):- connects(S0,a,S);
(8)  adjective(S0,S):- connects(S0,new,S);
(9)  common_noun(S0,S):- connects(S0,paper,S);
(10) common_noun(S0,S):- connects(S0,author,S);
```

The first clause means "a sentence extends from S0 to S if there is a noun phrase from S0 to S1, a verb from S1 to S2 and complements from S2 to S";

We use a 3-place predicate, 'connects', to represent terminal symbols in rules, where 'connects(S1,T,S2)' means "terminal symbol T lies between points S1 and S2 in the string". The above sentence is translated into the following set of unit clauses:

```
(11) connects(1,the,2);
(12) connects(2,new,3);
(13) connects(3,author,4);
(14) connects(4,writes,5);
(15) connects(5,a,6);
(16) connects(6,paper,7);
```

where each integer tags the points of the sentence as

follows:

"the new author writes a paper"
1 2 3 4 5 6 7

Note that the representation of a CFG by clauses is data-independent, because the actual representation of the string to be parsed is not known by the clauses. Another alternative for the representation is adopted in TUGA. It is characterized by tagging a point in a string, not by an integer, but instead by the list of symbols occurring after that point in the string. It is no longer necessary to provide a separate 'connects' clause for each symbol in the string. The first representation appears here for ease of explanation. In the case of Prolog, the proofs with the two representations will be essentially the same.

The goal of determining that the string of words is a sentence is expressed by the goal statement:

:- sentence(1,7).

which becomes the root of the proof tree. The string is in the language defined by the the grammar, iff the tree contains a halt statement (□), an empty assertion.

The parse is obtained by tracing the path from the root to the halt statement, and by noting at each step which clause was applied.

The following figure illustrates the parse tree for the sentence in question, from top-down and left-to-right, with four non-successful branches.

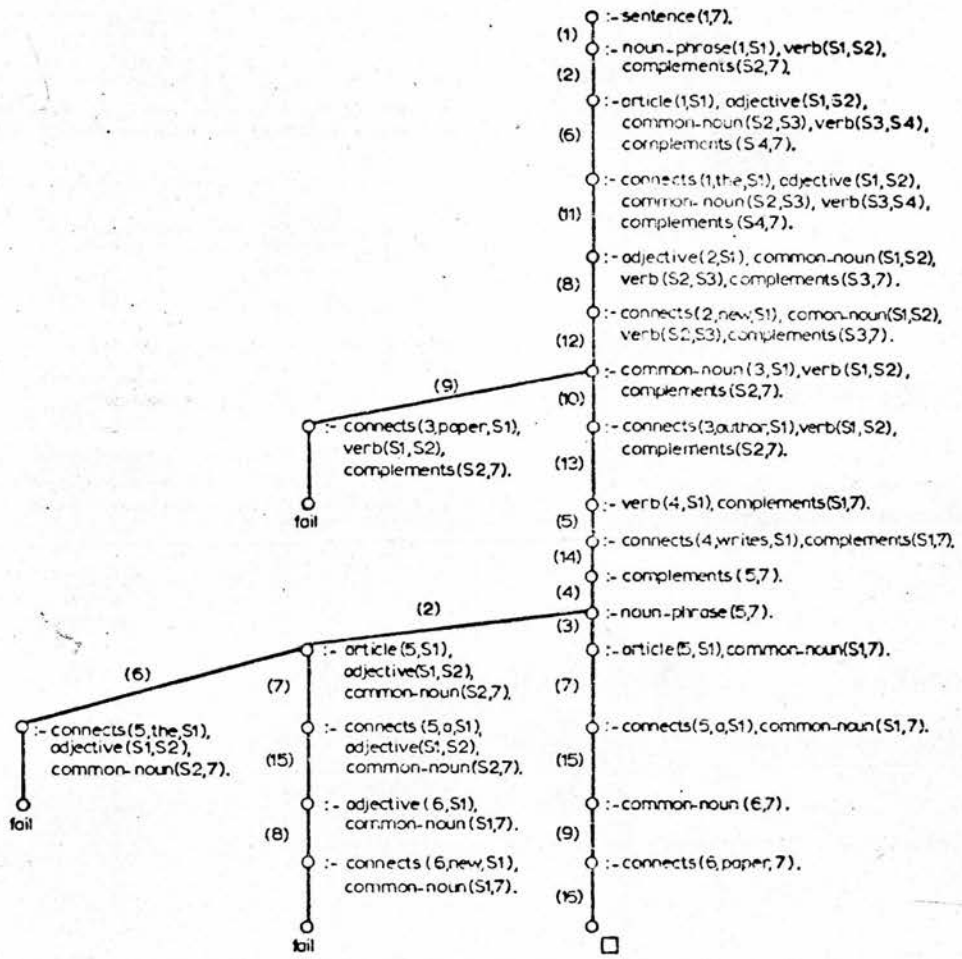


Fig.7

Parse tree

The parse tree shows the proof that clauses (1) to (16) plus the goal statement are inconsistent. This refutation is the generation of the parse. Note that any

derived goal statement Ci+1 is obtained from the preceding one Ci in the sequence by

(1) matching, with some substitution θ of the terms for variables, the first literal A in Ci with the literal A' in the conclusion (left-side of the clause) of some clause C of the set of clauses (1) to (16), whose variables have been renamed so as to be distinct from those of Ci;

(2) deleting the selected literal in Ci, and replacing it by the set of literals constituting the hypothesis (right-side of the clause) of C, and

(3) applying the matching substitution θ to the resulting clause.

Example 2: Let us consider a simplified grammar G which parses English sentences and produces their corresponding logical structures. We use now the notation of DCGs which is an extension of the notation used for context-free grammars. The grammar is defined by two modules: the syntax plus semantics (it corresponds to module GRAMMA of TUGA) and the morphology (it corresponds to module DICTIO of TUGA). For clarity, we have avoided the identifier abbreviations of the actual TUGA program.

Syntax plus semantics

```
sentence(S) --> noun_phrase(NP,S2,O),
                verb([subject-X,..L],O1),
                complements(L,O1,O2).
```

```
complements([],O,O) --> [];

complements([K-N|L],O1,O3) --> complements(L,O1,O2),
                               case(K),
                               noun_phrase(N,O2,O3);


noun_phrase(N,O2,O4) --> article(N,O1,O2,O3),
                         common_noun([subject-N|L],O1),
                         complements(L,O3,O4);

noun_phrase(PN,O,O) --> [PN], { proper_noun(PN) };

case(for) --> [for];
case(direct) --> [];
```

Morphology

```
verb([subject-A,for-P],is_published_by(A,P)) --> [writes];

common_noun([subject-P],publisher(P)) --> [publisher];

article(A,O1,O2,and(O1,O2)) --> [a];

proper_noun(hodges);
proper_noun(penguin);
```

Non-terminals are allowed to be compound terms in addition to the single atoms allowed in the context-free case. In the right-hand side of a rule, in addition to non-terminals and lists of terminals, there may also be sequences of procedures calls, written within the brackets '{' and '}'. These are used to express extra conditions which must be satisfied for the rule to be valid.

A non-terminal symbol is translated into an N+2 place predicate (having the same name), whose first N arguments are those explicit in the non-terminal and whose last two arguments are as in the translation of a

context-free non-terminal; procedure calls in the right-hand side of a rule are simply translated as themselves. For example, the rule

```
noun_phrase(PN,O,O) --> [PN], { proper_noun(PN) }.
```

represents the clause

```
noun_phrase(PN,O,O,S0,S):- connects(S0,PN,S),
                           proper_noun(PN).
```

The first grammar rule of G allows only for sentences comprising a noun phrase followed by a verb with possibly some complements. The first grammar rule for complements admits their absence (the terminal [] stands for the empty list), and the second rule defines the sequence of complements as a string composed by complement, a case and a noun phrase.

Let us explain the role of the logical variable in DCGs, a feature of logic programs. Different arguments of different non-terminals are linked by the same variable. This allows building up structures in the course of the unification process.

Consider for example the noun phrase "a publisher" which may be parsed and translated by the grammar rule,

```
noun_phrase(N,Oa,Ob) --> article(N,Oc,Od,Oe),
                         common_noun(N,Of),
                         {constraints(Oa,Ob,Oc,Od,Oe,Of)}.
```

Note that this rule is a simplified version of G's fourth rule. The non-terminal for a noun phrase has three arguments. The interpretation of the last argument Ob will depend on a property Oa of an individual N, because in general a noun phrase contains an article such as 'a'. The word 'a' has the interpretation Oe,

```
and(Oc,Od)
```

in the context of two properties Oc and Od of an individual N. The property Oc will correspond to the rest of the noun phrase containing the word 'a', and the property Od will come from the rest of the sentence. Therefore, Oe will contain an overall interpretation, and it is linked to Ob by the same variable. As Of is the property of the common noun, it is linked to Oc by the same variable. Oa has the description of the properties of N, and it will depend on the properties coming from the rest of the sentence. Therefore, Oa is linked to Od by the same variable.

Note that each grammar rule takes an input string, analyses some initial part, and produces the remaining part as output for further analysis.

Figure 8 illustrates the analysis tree for the sentence:

Hodges writes for Penguin

where, for each node step, is shown which grammar rule was applied by a label stating its non-terminal. Each node has a number corresponding to its depth. The analysis is done top-down and left-to right.

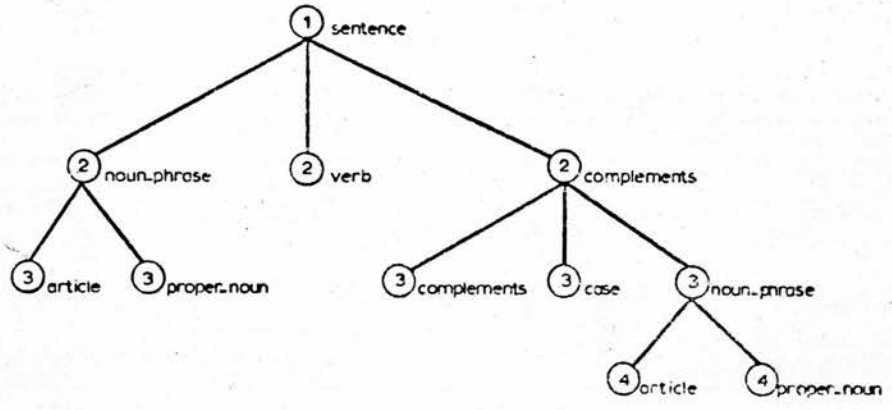


Fig.8

Analysis tree

Figure 9 illustrates the sequence of calls of the analysis of the sentence. The sequence is the scanning of a tree of grammar rules whose root is the goal -- the sentence to be analysed, and whose nodes are the sub-goals to be considered during the resolution of that goal. Each call is marked by two numbers and by the

name of the grammar rule used. The first number, placed at the left, designates the step order in a linear sequence of analysis steps. The second number designates the depth of search. The numbering is done top-down from the root, level 1. number corresponding to depth, is affected by a mark, '-' or '+', where the '-' means beginning of the resolution of a sub-goal, and the '+' means the successful conclusion of that resolution. For example, steps 3 and 4 represent two attempts at the resolution of the noun phrase 'hodges', using the grammar rules 'noun_phrase'. At step 3, the rule defining a noun phrase as an article followed by a common noun is tried. At step 4, the rule defining the noun phrase as a proper noun is tried. As 'hodges' is a proper noun, step 5 marks the success of the second attempt, and step 6 the success of the analysis of the noun phrase.

The argument numbers stand for variables, and the Prolog standard notation is used.

```
1  -1 sentence(91)
2  -2 noun_phrase(273,274,91)
3  -3 article(273,291,274,292)
4  -3 proper_noun(hodges)
5  +3 proper_noun(hodges)
6  +2 noun_phrase(hodges,91,91)
7  -2 verb([-(subject,hodges),...277],278)
8  +2 verb([-(subject,hodges),-(for,312),is_published_
      by(hodges,312))
```

```
 9  -2 complements([-(for,312)],is_published_by(hodges,
       312),91)

10  -3 complements(is_published_by(hodges,312),325)

11  +3 complements(is_published_by(hodges,312),is_publi
       shed_by(hodges,312))

12  -3 case(for)

13  +3 case(for)

14  -3 noun_phrase(312,is_published_by(hodges,312),91)

15  -4 article(312,351,is_published_by(hodges,312),352)

16  -4 proper_noun(penguin)

17  +4 proper_noun(penguin)

18  +3 noun_phrase(is_published_by(hodges,penguin),is_p
       ublished_by(hodges,penguin))

19  +2 complements([-(for,penguin)],is_published_by(hod
       ges,penguin),is_published_by(hodges,penguin))

20  +1 sentence(is_published_by(hodges,penguin))
```

Fig.9

Sequence of calls of the analysis

Step 20 is the final result of the analysis. The interpreted sentence was translated into the formula

is_published_by(hodges,penguin)

Figure 10 illustrates which input(SI) and output(SO) strings correspond to some step of the figure 9. Note that strings are represented as lists. For step 1 the input list is the sentence to be analysed and the output list represents the goal to attempt, ie. the empty list.

For step 6, while the input list is the sentence to be analysed, the output list hasn't the word 'hodges' which was already analysed.

```
Step

  1    SI = [hodges,writes,for,penguin]
       SO = [ ]

  2    SI = [hodges,writes,for,penguin]
       SO = 272

  3    SI = [hodges,writes,for,penguin]
       SO = 290

  6    SI = [hodges,writes,for,penguin]
       SO = [writes,for,penguin]

  7    SI = [writes,for,penguin]
       SO = 276

  8    SI = [writes,for,penguin]
       SO = [for,penguin]

  9    SI = [for,penguin]
       SO = [ ]

 10    SI = [for,penguin]
       SO = 324

 11    SI = [for,penguin]
       SO = [for,penguin]

 12    SI = [for,penguin]
       SO = 327

 13    SI = [for,penguin]
       SO = [penguin]

 14    SI = [penguin]
       SO = [ ]

 20    SI = [hodges,writes,for,penguin]
       SO = [ ]
```

Fig.10

Input and output strings of the analysis

3.4. EVALUATING THE LOGICAL STRUCTURE

The understanding of a sentence is the computation of a truth-value by evaluating its logical structure over a data base of facts and rules, according to a certain logical system, based upon predicate logic. The understanding of a question consists of finding instances of the corresponding declarative sentence which are true. The logical structure operates as a program whose meaning is given by rules of interpretation which provide its execution. The data base describes a certain situation in which the value of the logical structure, combined with the data items retrieved, supply an answer. The logical system adopted differs from predicate logic in that its quantifiers are more adjusted to the semantics of a natural language, the variables can denote both individuals and sets, and there is a third truth-value, undefined, besides true and false (see Appendix 1).

An example is presented to illustrate the processing behind evaluating a logical structure.

Example 1: consider grammar G presented in section 3.3 and a simplified algorithm for evaluating the logical structure. This algorithm is only the data base verification routine 'look_up'.

Data base verification

```
look_up(LS,true) :- LS,!.
```

```
look_up(LS,false).
```

Intensional data base

```
is_published_by(A,P) :- book(A,_,_,P,_).
```

Extensional data base

```
book(hodges,1,logic,penguin,1977).
book(bartlett,2,remembering,cambridge,1932).
book(culicover,3,syntax,academic,1977).
```

The evaluation of the logical structure (LS) is carried out by the Prolog proof procedure, and consists of showing whether or not LS is logically entailed by the data base. An answer is derived according to the result of that evaluation. If a proof is found, LS is a logical implication of the data base and a 'yes' answer is produced. If no proof is found, the goal 'LS' fails and the completeness (in principle) of the Prolog proof procedure justifies the 'no' answer.

The figure below shows the sequence of calls for the query,

Does Hodges write for Penguin?

The notation is similar to that adopted in section 3.3, and parsing calls were not considered in the following figure,

```
   .
   .
   .
-4  pn(hodges)
+4  pn(hodges)
   .
   .
   .
-4  pn(hodges)
+4  pn(hodges)
   .
   .
   .
+2  prop(pub_of_aut(penguin,hodges))
   .
   .
   .
-3  look_up(is_published_by(hodges,penguin),400)
-4  is_published_by(hodges,penguin)
-5  book(hodges,401,402,penguin,403)
+5  book(hodges,1,logic,penguin,1977)
+4  is_published_by(hodges,penguin)
+3  look_up(is_published_by(hodges,penguin),true)
   .
   .
   .
Yes.
   .
   .
   .
```

Fig.11

Sequence of calls for a query

The query was translated into the following formula, which is used as a goal

```
is_published_by(hodges,penguin)
```

The result is true, and thereby the response 'Yes.' is delivered to the user.

We consider briefly two algorithms for evaluating the logical structure, before explaining our own. The one proposed by (Colmerauer,1977) consists of decomposing the logical structure into elementary formulae which are evaluated according to the semantics of a logical system. Evaluation is done upon sets of individuals, following the hypothesis: "the n-ary properties introduced by verbs, nouns and adjectives apply to sets of individuals". The other algorithm implemented by (Dahl,1977) follows closely Colmerauer's proposal, and is supported on the concept of domain, viewed as a set of individuals. The data base, a collection of explicit domains, is not of a relational type.

Our approach to evaluating the logical structure rests upon the modification of the scope of the above hypothesis, and on the organization of the data base.

Colmerauer's hypothesis was re-written as follows: "the n-ary properties introduced by verbs, nouns and adjectives apply to individuals and to sets of individuals". The formulation of this hypothesis is closely related, but differs from Colmerauer's. From a linguistic point of view, the individual is considered as a set with one member, just as Colmerauer suggests. From

a computational point of view, individuals are selected from the data base one by one, and not as a set, and relations over sets are verified by testing them on all possible combinations of values in the argument positions.

Colmerauer considers that properties do not apply to individuals but to sets of individuals, and he argues with two reasons:

- not all properties of sets can be reduced so easily into properties ranging over individuals, especially those involving notions of cardinality; and
- a property on individuals can always be expressed in terms of a property on sets by representing an individual as a set of individuals with cardinality 1.

These reasons were adopted by Dahl who added a semantic advantage regarding the implementation of the Colmerauer's framework:

- the possibility of organizing the world as an hierarchy of domains, where each is a set of individuals.

The introduction of the concept of domain has, according to Dahl, three consequences:

(1) reference, as regards searching for data, is done to domains and not to individuals;

(2) a better definition for the arguments of a relation (indexing on domains may support a faster searching);

(3) a solution for the negation problem (negation is viewed as the complement of a

set).

These arguments are debatable. We note that Colmerauer's hypothesis depends on the kind and organization of the world considered, and not on cardinality. The open world recommends a hypothesis based on sets, and the closed world a hypothesis based on individuals. The open world is defined as the one where the only answers to a query Q are those which are obtained from proofs of Q given a data base as hypothesis. The closed world is defined as the one where certain answers are admitted as a result of failure to find a proof. The different points of view are brought up by the nature of the world. In an open world, the data base is viewed as a theory, because it is constructed as the new incoming facts arrive: the appearance of a property opens a set of individuals, and only the subsequent dialogue specifies their nature (Pasero,1976). In a closed world, the data base is viewed as an interpretation, because the individuals are completely specified (Coelho,1979a).

The experimentation with our program TUGA, working within a closed world, proved that for simple sentences like

Artificial Intelligence is a book by Nilsson.

the reference to domains, considered as explicit sets of

individuals, provokes a large searching process which increases when the facts of the data base increase. This is precisely the opposite to what is suggested by (Dahl,1977), but her data base was so small that the fact was not obvious. This finding justifies that domains should be defined implicitly as relational properties and that their arguments may refer either to single individuals or to sets of individuals. This option involves a verification of properties and a treatment of negation that differs from Dahl's, and we discuss them further on in Chapter 4.

Example 2 : consider the following sentence;

No publisher has an author.

which is translated into the logical structure, represented by the tree of figure 12.

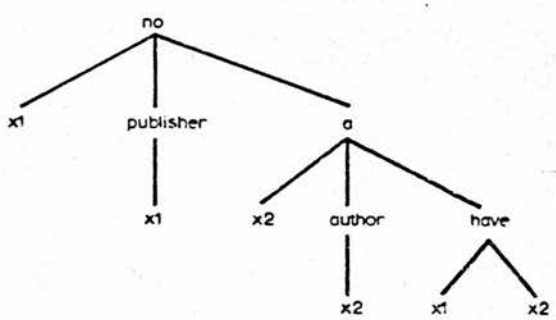


Fig.12

Tree structure

The logical structure, as produced by program TUGA, is the following one:

```
not(
   for(X1;
       and(publisher(X1);
           for(X2;
               and(author(X2);
                   have(X1,X2));
               card(X2;greater,0)));
       card(X1,greater,0)))
```

The logical structure is composed of well-formed formulae of our logical system, and is represented as a tree structure with seven blocks, as shown in figure 13. Note the presence of predicates 'not', 'for', 'and', 'card' and 'greater'. Their combination supports the translation of the meaning of articles 'no' and 'a', according to the definitions presented in Appendix 1.

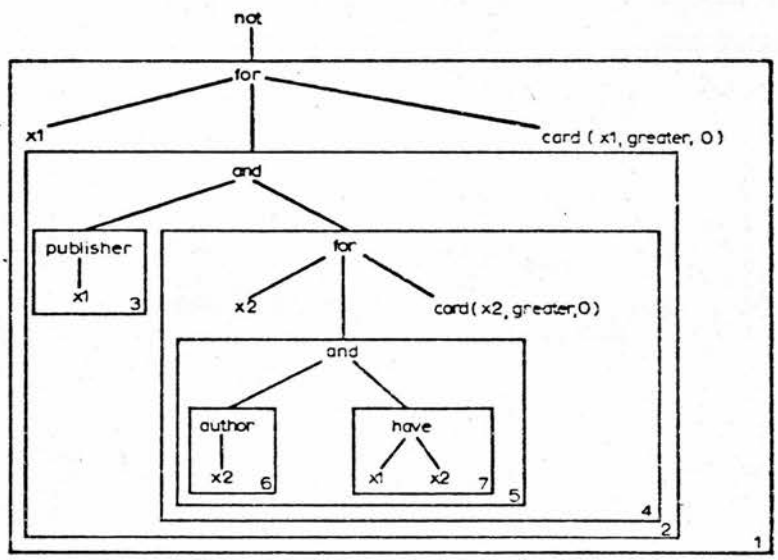


Fig.13

Block tree structure

The evaluation of the overall logical structure is the computation of its meaning by instantiating its variables and calculating its truth-value. It is done

top-down and from left-to-right, as follows:

(i) The truth-value of the structure is the negation (see the negation table in Appendix 1) of the truth-value of block 1 (dominated by 'for');

(ii) The truth-value of block 1 is obtained by evaluating block 2 (dominated by 'and'); and by checking whether its result is compatible with the cardinality definition of the quantifier standing for 'no';

(iii) The evaluation of block 2 consists of all the possible instances of block 3 over the data base, and for each of them, the evaluation of block 4 (dominated by 'for');

(iv) The truth-value of block 4 is obtained by evaluating block 5 (dominated by 'and'), and by checking whether its result is compatible with the cardinality definition of the quantifier standing for 'a';

(v) The evaluation of block 5 consists of all the possible instances of block 6 over the data base, and for each of them, the resolution of block 7 over the data base;

(vi) The value of 'X1' is the list of all publishers who have authors; and the value of 'X2' is the list of these authors;

A final note on the logical system adopted, which is based upon a three-valued logic. The motivation for such a system was given by the inadequacy of classical logic for treating certain articles, such as the definite article 'the'. The definition of truth of this article, represented as a three-branched quantifier 'the(x,P,Q)', depends on two conditions of truth: one about the set X of x's satisfying the property P; and the other about the property Q specifying the domain of x's.

val[the(x,P,Q)]=true if X={x ¦ P} is a unit set
and val(Q)=true

val[the(x,P,Q)]=false if X={x ¦ P} is a unit set
and val(Q)=false

val[the(x,P,Q)]=undefined if X={x ¦ P} is an empty set
or a multiple element set
or val(Q)=undefined

Note that the third value undefined is only generated if P is false or Q undefined.

However, our definitions of Portuguese articles, like Colmerauer's and Dahl's, are weak as regards the generation of the third value undefined. Only the definitions for articles 'the' and 'the i' may cover meaningless situations, because a conditional is present. Undefined appears when the cardinality is false or the value of statement 'e2', corresponding to formula f2 in q(x,f1,f2), is undefined. A new definition of articles, proposed by (Pique,1978), overcomes the weak treatment of meaningless. He considers these definitions based upon

the straightforward use of complex quantifiers, like 'a(x,P,Q)', and that 'P' represents a presupposition. The value undefined for 'a' is generated when the value of 'P' is false and the value of 'Q' undefined.

The logical system adopted is more restricted than, for example, the three-valued system of (Lukasiewicz,1966), because it represents a compromise between formal logic and the linguistic meaning of logic. The logical operations such as 'and', 'if, 'not', 'equal' and 'greater' are only defined in the semantics of closed statement formulae, which are the possible single components of the logical structure of a natural language sentence. The logical tables are constructed with a three-valued logic and a total ordering, which has not the value undefined between the other two values, as it is usually adopted for logical systems having conjunction and disjunction. A reason for this is the compromise stated above on adequacy criteria. The tables show two facts: 1) the operator 'if' is formalised as a linguistic presupposition rather than as the usual implication, therefore it can not be defined upon 'and' and 'not', and 2) de Morgan's laws are not verified. In fact, there is no disjunction defined as a maximum between truth values, because it is not clear how logical disjunction can stand for linguistic disjunction. We observe also that the operation 'if' is different from the presupposition considered by (Keenan,1972). But the

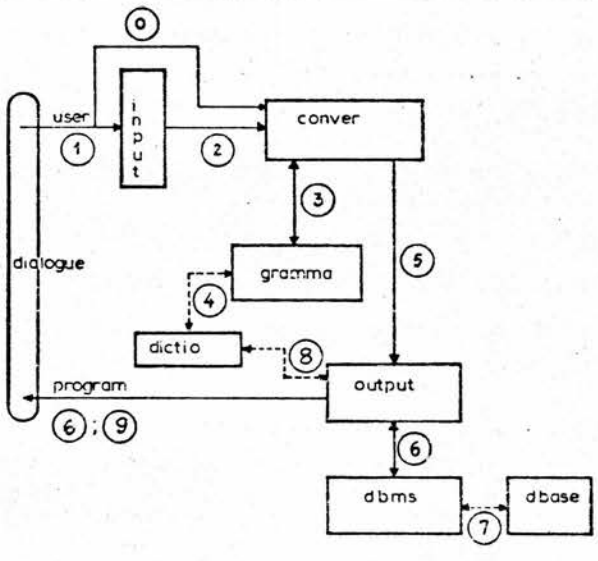
tables for negation and conjunction are similar to Keenan´s ones.

CHAPTER 4

IMPLEMENTATION

4.1. ORGANIZATION OF THE PROGRAM

TUGA is a program organized into modules. This modularity ensures that some modules remain the same when the subject matter changes, and that a module may be improved without affecting the others or the design as a whole.



NB. Circled numbers point the direction of the natural language processing

Fig.14

Modular organization of program TUGA

TUGA consists of seven modules: INPUT, CONVER, GRAMMA, DICTIO, DBMS, DBASE and OUTPUT (see Appendix 3).

Module INPUT handles the natural language input sentences by transforming the sequence of written words and punctuation marks into lists of words and marks, and it checks whether every word is known.

Module CONVER is the supervisor, ie. the high level control structure of all processes occurring during a dialogue. It is called by users and it processes all the exchanges, either by taking the initiative or by giving it to them: a grammar of dialogues is responsible for all kinds of interactions between the program and its users. It calls the module GRAMMA for analysing the sentence, deciding upon its result to accept or to refuse it. In the case of accepting it, it calls the module OUTPUT to interpret it, and to prepare the appropriate answer, remark or question.

Module GRAMMA handles a definite clause grammar (DCG). It takes a list, representing the natural language input sentence and applies several knowledge sources to yield a logical structure that contains all the information for a semantic interpretation. It is able to extract from a set of possible readings the appropriate reading. The grammar, being able to generate all and only the sentences of a subset of Portuguese, recognizes if the input sentences belong to that subset,

and it establishes a relation between the written sentence and its meaning.

Module DICTIO contains the language specific knowledge about the written input and output sentences, ie. a dictionary with the required lexical items to support common dialogues in the library world.

Module DBMS evaluates the logical structure by decomposing it into single components. Each one is verified by consulting the data base either to find or to relate the individuals belonging to the arguments of each corresponding relation.

Module DBASE contains domain-specific knowledge --the library, defined by the explicit facts about the document collection and the rules for deriving the implicit facts, and the classification system--and ability to make deductions and connect facts in the subject domain, such as the classification method.

Module OUTPUT handles the formation of answers, remarks or questions to the user. It calls the module DBMS to evaluate the logical structure, and according to which it chooses and it generates the appropriate output form.

4.2. THE DATA BASE AND ITS ORGANIZATION

The library data base (module DBASE of program TUGA), is a closed set of objects regarding our document collection, and formally represents the library world. It is a virtual relational data base, represented in predicate logic. It is relational because data items are organized as sets of similar tuples of items, where each set is a relation and access can be made through any combination of items. It is a virtual data base because it also consists of rules {1}. A rule defines a relation which is not present as a set of tuples, and therefore operates as a data base access function by providing the possibility of answers which are not explicitly present. The collection of sets of similar tuples is called the extensional data base, and the collection of rules is called the intensional data base.

We use the clausal form of first-order predicate logic as the abstract representation of a virtual relational data base. Terms of logic represent objects; the unit clauses represent facts and non-unit clauses the rules.

{1} The relational data base according to (Codd,1972) consists of sets of similar tuples only, and it is an actual relational data base.

The elementary facts or objects of the library world that constitute the extensional data base are organized in predicate logic as sets of unit clauses. There are two sets: one for books (relation 'book') and another for papers (relation 'paper').

Example: the 7-ary relation 'book' is written in Prolog as a set of unit clauses,

```
book(winston,1,'psychology of computer vision','mcgraw
        hill', 1975,[1214,1216,1222],[3,4]).
.
.
.
```

This unit clause has the following reading:"Psychology of Computer Vision", book no.1 of the collection, was written by Winston and published by "McGraw Hill" in 1975; it is classified under categories 1214, 1216 and 1222, and has documents nos.3&4 as its bibliography.

The arguments of the 7-ary relation 'book' corresponding to the author, number, title, publisher and year are constants (ie. a single value for a particular domain). The other two arguments corresponding to the classification and bibliography are sets of constants (ie. lists of values).

This relation could be represented as the following table;

book

aut		tit	pub	yea	cat	ref
winston	1	psycology of computer vision	mcgraw hill	1975	1214, 1216, 1222	3, 4
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig.15

Description of 'book'

The objects of the library closed world are classified into categories, called domains. A domain is essentially a relation providing a way to access a certain set of objects, one by one. Each domain is identified by three arguments; its name, its hierarchical representation and a variable for receiving the result of the access to the relation where the objects are stored. The variable, by receiving each member of the set, may be considered as the implicit representation of the set.

The relationships of the identified domains for the library world are organized into a tree data as shown in figure 16.

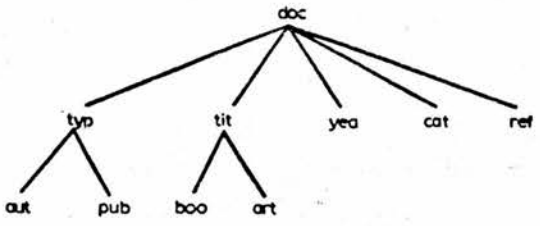


Fig.16

Hierarchy of domains

where 'doc' means document, 'typ' type, 'boo' book, 'art' paper, 'aut' author, 'tit' title, 'pub' publisher, 'yea' year, 'cat' category and 'ref' reference.

Example: the domain of authors is defined in Prolog by its declaration

```
domain(aut,aut([]),A) :- author(A).
```

and by the auxiliary relations,

```
author(A) :- author(A,_).
author(A,T) :- book(A,_,T,_,_,_,_);
               paper(A,_,T,_,_,_,_).
```

N.B. Anonymous variables are written in Prolog '_'.

Such a definition of a domain allows the implicit identification of its members through the variable 'A'. The term 'aut([])' defines a fragment of the hierarchy of domains by meaning that below node 'aut' there is [], ie. no other node. For the case of a domain placed higher in the hierarchy tree, the hierarchy representation may or may not contain the description of the tree from that node to the terminal node.

Example: the domain of the library world types has the following hierarchy representation:

'typ(V)'

where 'V' means a variable to be instantiated either with authors, 'aut([])' or publishers, 'pub([])', ie.

'typ(aut([]))' or 'typ(pub([]))'

The organization of objects into domains, and the subsequent arrangement of these domains into a hierarchical structure, is a device for constraining a broad class of individuals. This implements the notion of a variable restricted in range to some set of values, or to values having some property. The system of domains is a device to make a distinction between different sorts of individuals. It acts as a guide for the further data base search and as a filter for the rejection of syntactically ungrammatical and semantically anomalous strings.

The rules are essential for accomplishing data base queries which correspond to the interpretation of natural language requests.

Example: the 2-ary relation 'published_after' is written in Prolog as the following set of 2 clauses:

```
published_after(T,Y1):- year_of(T,Y), Y>Y1.

year_of(T,Y):- book(_,_,T,_,Y,_,_);
               paper(_,_,T,_,Y,_,_).
```

with the following reading:"'T' is a document title published after the year 'Y1' if there is a document with title 'T' published in the year 'Y' which is greater than the year 'Y1'".

In the query,

"Quais os livros publicados apos 1975?
(What are the books published after 1975?)

the word 'publicados' (published) introduces the 2-ary relation 'published_after' where the argument corresponding to title(s) is a variable, and the argument corresponding to year(s) is a constant and equal to '1975'. The query is interpreted as the search for all possible titles (books) published after 1975. The variable gets instantiated with the possible set of titles (the empty list, one or more individuals).

In the figure 17 is shown a fragment of the organization of the library data base with special emphasis on domains, objects and relations (see the glossary of abbreviations used in program identifiers in Appendix 5).

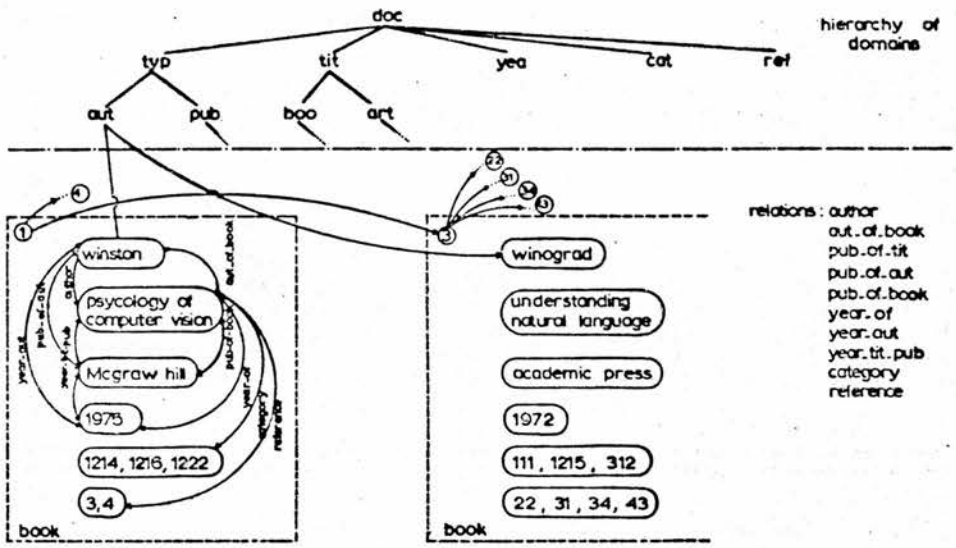


Fig.17

A fragment of the library world:
domains, objects and relations

4.3. THE KNOWLEDGE BASE AND THE CLASSIFICATION METHOD

The knowledge base (module DBASE of program TUGA) is the set of processes and associated data structures which provide the program with a particular kind of information and with reasoning. This information deals with domain-specific knowledge, the classification system for AI and the classification method. Reasoning covers the classification method implementation, the ability to generate new categories for the classification system, the storage of new documents by asking for the missing data items, and some support machinery for running dialogues.

An essential feature of TUGA is that every fact about a document (eg. its author) or the name of a new category supplied by the user is represented in a standard way. Unit clauses, such as 'book' and 'paper' for documents and 'cat' for categories, are created and added to the existing data base.

Example: the category 'natural language systems' is defined in Prolog by the unit clause:

```
cat('artificial intelligence applications',
    121,
    'natural language systems',
    1215).
```

with the following meaning: "the category 'natural language systems' has the number 1215, and is an 'artificial intelligence application'".

Higher level structures, such as those which classify and generate categories, are built by Prolog clauses referencing this uniform knowledge base. These structures are called by the overall control structure, and according to the grammar of dialogues, cause questions to be asked about missing facts. For example, during the process of classifying a document, the structure classify is only called after obtaining all facts (titles of the documents referenced) known by the user. It handles this information by eliminating redundant categories, and by proposing a list of categories for the classification of the document. But,

in case of rejection, the structure engages an interaction with the user by asking new questions. If the user proposes categories by himself, a test is made to check whether they are known to the classification system. New categories can therefore be created by the structure to generate categories, and can be archived.

4.4. THE FRAGMENT OF PORTUGUESE GRAMMAR

The linguistic part of TUGA is formalised as a DCG which handles a subset of Portuguese syntax and vocabulary relevant to the domain of dialogues in a library environment.

The grammar satisfies the two main criteria for being an adequate model of an interesting subset of Portuguese. Firstly, it is able to distinguish the set of well-formed sentences from its complement. Secondly, it associates an adequate logical structure with each sentence.

In this section we describe the aspects to be considered in writing the DCG for a subset of Portuguese, covering current dialogues in a library world.

First, the dialogues are analysed in order to bring out the kinds of linguistic constructions involved and the basic syntactical categories. Second, the words

occurring in the dialogues are classified into two groups: the core vocabulary (articles, prepositions, adverbs, etc.), which are domain independent and useful for all other domains of dialogues; and the peripheral vocabulary (nouns, verbs and adjectives), which are domain dependent. Third, the words are defined by taking into account their participation (function, form and meaning) in the dialogues. Fourth, the regularities observed in the linguistic constructions--how they are formed and understood--are synthetized into grammar rules, which correctly recognise the structure of Portuguese sentences.

4.4.1. BRIEF SURVEY OF PORTUGUESE GRAMMAR

It is useful to present Portuguese grammar by briefly surveying the main features of its syntactic component, leaving the arguments and the details to the next subsections, where its implementation is discussed. In doing so, we also present the sort of sentences occurring in the library world. The main words, associated to each feature, are underlined.

1) Variety of linguistic constructions

statement: "Artificial Intelligence" e´ um livro de Nilsson o qual foi publicado em 1971.
("Artificial Intelligence" is a book by Nilsson which was published in 1971.)

question: Quantos artigos existem?
(How many papers are there?)

command: Classifique "Artificial Intelligence"!
(Classify "Artificial Intelligence"!)

2) Negative interrogative forms

Nao escreve nenhum autor para a "Academic Press"?
(Doesn´t any author write for "Academic Press"?)

Nao escreve ninguem para todos os livreiros?
(Doesn´t nobody write for all publishers?)

Quem nao escreve para todos os livreiros?
(Who does not write for all publishers?)

3) Double negation imposed by the use of ´nenhum´ (no)

Nilsson nao tem nenhum livreiro.
(Nilsson doesn´t have any publisher.)

Nilsson nao escreve para nenhum livreiro.
(Nilsson doesn´t write for any publisher.)

Elsevier nao e´ o livreiro de nenhum autor.
(Elsevier is not the publisher of any author.)

Elsevier nao tem o titulo de nenhum autor.
(Elsevier doesn´t have the title of any author.)

4) Coordination of statements and questions

statements: Nilsson escreveu a "Artificial Intelligence", e
"McGraw Hill" e´ o livreiro de Nilsson.
(Nilsson wrote "Artificial Intelligence", and
"McGraw Hill" is the publisher of Nilsson.)

questions: Quais sao os autores conhecidos, e
qual o livreiro de Nilsson?
(Who are the known authors, and
what is Nilsson´s publisher?)

5) Coordination of proper nouns

subjects: Nilsson e Charniak sao autores.
(Nilsson and Charniak are authors.)

complements: Os livreiros dos autores sao a "McGraw Hill" e a Elsevier.
(The publishers of the authors are "McGraw Hill" and Elsevier.)

prepositional complements: Os autores dos livros estao publicados na Freeman e "North-Holland".
(The authors of the books are published by Freeman and "North-Holland".)

6) Coordination of common nouns and adjectives

nouns: Quais sao os autores e livreiros conhecidos?
(Who are the known authors and publishers?)

adjectives: Quais sao os autores conhecidos e classificados?
(Who are the known and classified authors?)

7) Disjunction of proper nouns

complements: Quais os artigos de Warren ou Kowalski?
(What are the papers by Warren or Kowalski?)

prepositional complements: Os autores dos livros foram publicados pela Elsevier ou "Academic Press"?
(The authors of the books were published by Elsevier or "Academic Press"?)

8) Complements

phrase complements: Warren e´ um autor referenciado.
(Warren is a referenced author.)

adjective complements: Warren esta´ referenciado.
(Warren is referenced.)

prepositional Nilsson esta´ publicado pela "McGraw

complements: Hill".
(Nilsson is published by "McGraw Hill".)

9) Relative/interrogative pronoun 'quem' (who)

used only with nouns that refer to humans,

Quem e' o autor de "Artificial Intelligence"?
(Who is the author of "Artificial Intelligence"?

10) Special verbs

'ser' : Nilsson e' um autor.
(Nilsson is an author.)

'estar' : "Artificial Intelligence" esta' referenciada.
("Artificial Intelligence" is referenced.)

'ter' : Qual e' o titulo que Nilsson tem?
(Which is the title that the book by Nilsson has?)

'haver' : Ha' algum artigo escrito por Warren?
(Is there any paper written by Warren?)

11) Verb tense ellipsis('to be')

'e'' : Qual (e') o autor de "Computational Semantics"?
(Who is the author of "Computational Semantics"?)

12) Subject agreement

A "Artificial Intelligence" e' conhecida.
("Artificial Intelligence" is known.)

13) Paraphrase power

Em que ano foi publicada a "Artificial Intelligence"?

(In what year was "Artificial Intelligence" published?)

Quando foi publicada a "Artificial Intelligence"?
(When was "Artificial Intelligence" published?)

Qual o ano de publicacao de "Artificial Intelligence"?
(What is the date of publication of "Artificial Intelligence"?)

14) Quantificational power

Ninguem escreveu a "Artificial Intelligence".
(Nobody wrote "Artificial Intelligence".)

Qualquer autor escreve para a Freeman.
(Any author writes for Freeman.)

Nem todos os autores escrevem para "Academic Press".
(Not all authors write for "Academic Press".)

Todos os autores nao sao livreiros.
(All the authors are not publishers.)

Cada livro tem um autor.
(Each book has an author.)

Back nao tem nenhum livreiro.
(Back has no publisher.)

Alguns autores escrevem para Elsevier.
(Some authors write for Elsevier.)

Alguem escreve para a "Academic Press".
(Somebody writes for "Academic Press".)

Os cinco autores estao referenciados.
(The five authors are referenced.)

Os autores referenciados sao cinco.
(The referenced authors are five in number.)

15) Adverbial use

statement: "Computational Semantics" nao tem a "North-Holland" como livreiro.
("Computational Semantics" doesn't have "North-Holland" as publisher.)

question: Como classifica a "Computational semantics"?
(How do you classify "Computational semantics"?)

16) Proper nouns with definite article

O Nilsson e´ o autor da "Artificial Intelligence".
(Nilsson is the author of "Artificial Intelligence".)

17) Prepositional phrases

with noun-phrase: Quais sao os livros sobre Prolog?
(What are the books about Prolog?)

open question: Por quem e´ publicada a "Artificial Intelligence"?
(By whom is "Artificial Intelligence" published?)

4.4.2. PORTUGUESE LINGUISTIC CONSTRUCTIONS

The linguistic constructions handled by TUGA are divided into two main groups: compound and single constructions.

Compound constructions are a sequence of single ones, to form multiple statements, questions or commands, and sequences linked by a separator (eg. comma, conjunction ´e´ (and)) to form coordination of statements. They are available to users.

Single linguistic constructions--declarative, interrogative and imperative sentences--are used either by the program or by its users in affirmative or negative form.

Let us consider the possible linguistic constructions, starting with the user types: requests (statements, questions and commands) and answers.

I) Types of user statements

User statements are declarative sentences performing activities of asserting and denying.

User statements are classified into single or compound statements.

Example of compound constructions:

- Lehnert escreveu "The process of question answering", e Charniak escreveu "Computational Semantics".
  (Lehnert wrote "The process of question answering", and Charniak wrote "Computational Semantics".)

Example of single constructions:

- Bach nao tem nenhum livro.
  (Bach has no book.)
- A "Artificial Intelligence" e´ um livro do Nilsson que foi publicado em 1971.
  ("Artificial Intelligence is a book by Nilsson that was published in 1971.)

II) Types of user questions

User questions are classified as single or compound, and are grouped according to their linguistic form and interrogative pronoun into two sets: closed and open questions. Closed questions merely require a yes or no answer, while open questions require one or more phrases as answer.

Examples:

Open: Quem e´ o livreiro da "Introduction to Logic"?
(Who is the publisher of "Introduction to Logic"?)

Closed: E´ "Human problem solving" um livro?
(Is "Human problem solving" a book?)

Closed: Sera´ que Back e´ um livreiro?
(Is it the case that Back is a publisher?)

The following table presents the types of questions the user can put.

TYPES OF USER QUESTIONS

main type	secondary type	initial marker type	portuguese marker	english marker
closed	disjunctive			
	confirmation/ denial			
open-ended		which	que o que quem prep que prep quem quanto onde quando como quais	(what, whom) (what) (who; whom) (prep what) (prep who) (how much) (where) (when) (how) (which; what)
		howmany	quantos quantas	(how many) (how many)
		pre-locutory expressions	Por favor... Será que... Diga-me que... Gostaria de saber se...	(Please...) (Is it that...) (Tell me that...) (I whould like to know if...)

N.B. closed questions = confirmation/denial seeking
open ended questions = information seeking

Fig.18

Types of user questions

III) Types of user commands

User commands are imperative sentences performing activities of requesting information and/or of posing problems.

Examples:

- Crie uma categoria!
  (Create a category!)
- Informe-me sobre "Psychology of Computer Vision"!
  (Inform me about "Psychology of Computer Vision"?)

The following table presents the types of user commands:

TYPES OF USER COMMANDS

type	portuguese	english
pre-locutory expression	De-me... Diga... Diga-me... Informe-me...	(Give me...) (Say...) (Tell me...) (Inform me...)
order	verb (imperative) ...	

Fig:19

Types of user commands

IV) Types of user answers

User answers are declarative sentences, in the affirmative or negative form, for guiding the program in choosing the dialogue models or exchange forms, or for giving data (eg. the title of a document) that it uses for deriving the final answer (eg the document classification):

The following table presents the types of user answers:

TYPES OF USER ANSWERS

general type	particular type	portuguese sentence	english sentence
quiescent	refusal	não.	(no.)
		nao está bem.	(it is not all right.)
		não quero.	(I don't want.)
		não concordo.	(I don't agree.)
		não conheço.	(I don't know.)
		não tenho nenhum.	(I have none.)
		não tenho mais nenhum.	(I haven't got any more.)
	acceptance	sim.	(yes.)
		está bem.	(all right.)
		sim quero.	(yes, I want it.)
		quero.	(I want it.)
		sim concordo.	(yes I agree.)
		concordo.	(I agree.)
		sim conheço.	(yes, I know.)
		conheço.	(I know.)
		por favor.	(please.)
		ok.	(ok.)
		é.	(it is.)
	hanging-up	deixe	(leave it.)
		desisto	(I give up.)
		quero mudar de conversa	(I want to change conversation.)
		vamos mudar de conversa	(let us change conversation.)
		mudo de conversa	(I change conversation.)
		mudo de diálogo	(I change the dialogue.)
	want	quero... .(I want...) (N objects, where N is an integer and objects are documents (titles) and categories)	
		quero mais... (I want more...) (N objects)	
	goodbye	adeus.	good bye
effective	titles		
	conjuction of nouns	(nâmes: proper nouns; common nouns and integers (category or year)	

N.B. quiescent answers = information guiding
effective answers = information giving

Fig. 20

Types of user answers

Let us now consider the possible linguistic constructions generated by the program:

V) Types of program questions

Program questions are pre-defined interrogative sentences with arguments, instantiated by user answers or by the conversational scenario. They are classified in

the following way:

TYPES OF PROGRAM QUESTIONS

particular scenario	portuguese sentence	english sentence
information output	E que mais ? Quer mais ?	(And what else ?) (Do you want any more?)
document classification	Por favor, dê-me o título de uma (outra) referência do documento em questão	(Please, give me the title of a (another) reference to the document in discussion)
	Por favor, faça a sua escolha tendo em atenção as categorias sugeridas e as que julgar mais apropriadas. Dê-me 3 categorias no máximo.	(Please, choose according to the suggested categories and the most appropriate ones. Give me a maximum of 3 categories.)
	Quer classificá-lo ?	(Do you want to classify it ?)
	Dê-me 3 categorias no máximo!	(Give me a maximum of 3 categories!)
	Por favor, conhece a classificação de ——?	(Please, do you know the classification of ——?)
classification category generation	Por favor, debaixo de que categoria a pretende inserir?	(Please, under what category do you want to insert it?)
	Por favor, qual o nome da nova categoria ?	(Please, what is the name of the new category ?)
addition and/ /or deletion of data items	Por favor, qual o nome do —— do documento ?	(Please, what is the name of —— of the document ?)
	Por favor, qual é —— do documento ?	(Please, what is —— of the document ?)
	Por favor, deseja arquivar este documento na Base de Dados ?	(Please, do you want to store this document in the Data Base ?)
	Por favor, quais são as —— do documento ?	(Please, what are —— of the document ?)
information transaction	Existe algum erro sintático na escrita da sua frase ?	(Is there any syntactic failure in the writing of your sentence ?)
	A palavra desconhecida é um nome próprio ?	(Is the unknown word a proper noun ?)
	Qual a palavra que é nome próprio ?	(Which is the word that is a proper name ?)
	Qual é o género de —— ?	(What is the gender of —— ?)
	A qual dos tipos, autor, título, livreiro, ou categoria, pertence a palavra —— ?	(To which of the types, author, title, publisher, or category, does the word —— belong ?)
	Vou perguntar-lhe informações sobre —— de forma a inseri-la no dicionário. Concorda ?	(I am going to ask you information about —— in order to insert it in the dictionary. Do you agree ?)

Fig. 21

Types of program questions

VI) Types of program answers

Program answers are pre-defined declarative sentences with arguments, instantiated by user answers or questions, or the conversational scenario. They are classified in the following way:

TYPES OF PROGRAM ANSWERS

general type	particular type	portuguese sentence	english sentence
quiescent	refusal	Não. Não concordo. Não compreendo a frase. Nenhuma entidade satisfaz a sua pergunta. A sua pergunta é indefenida. Não encontro nenhuma entidade que a satisfaça. A sua frase pressupõe outros factos, logo um contexto. Como não possuo informação sob o que foi dito anteriormente, a sua frase é ambigua, portanto não consigo responder-lhe. Não compreendo esta frase porque a palavra —é desconhecida.	(No.) (I don't agree.) (I don't understand the sentence.) (No entity satisfies your question.) (Your question is unclear.) (I don't find any entity that satisfies it.) (Your question presuppose other facts, therefore a context. Your sentences is ambiguous because I have no information about what was said previously. Therefore I cannot answer you.) (I don't understand your sentence because the word is unknown.)
	acceptance	Sim. Concordo. Olá. Vamos iniciar a conversa. Por favor, escreva factos ou perguntas. Olá. Vamos retomar a conversa anterior. Por favor, escrev factos ou perguntas. Ok, esta conversa terminou. Adeus e até à vista Muito obrigado. Está bem. Por favor, preste atenção à palavra que escreveu.	(Yes.) (I agree.) (Hello, Let us start the conversation. Please, enter statements or questions.) (Hello, Let us restart the previous conversation. Please, type facts or questions.) (Ok, this conversation ended. Goodbye and see you soon). (Thank you very much) (All right.) (Please, look carefully to the written word.)
	hanging-up	Desviá-mo-nos da conversa! Ok. Vamos desviar-mo-nos da conversa!	(We changed conversation!) (Ok. Let us change conversation!)
effective	conjunction of names		
	existance	Existem mais N!	(There are N more!) N is an integer

NB. quiescent answers = information guiding
effective answers = information giving

Fig.22

Types of program answers

4.4.3. BASIC SYNTACTICAL CATEGORIES

Syntactical categories are the primitive concepts from which syntactical rules are constructed. We present the basic categories used to compose general Portuguese sentences.

The basic syntactical categories are: noun and verb phrases, articles, nouns, verbs, adjectives and their complements, and relative clauses.

A) Noun and verb phrases

A1) A noun phrase has the following form:

A11) a sequence of proper nouns connected by 'e' (and), 'ou' (or) and comma

Examples:

- ... Hewitt, Winograd e Nilsson...

- ... Warren ou Pereira...

A12) an article (sometimes implicit), a sequence of adjectives, a common noun, possibly a sequence of adjectival groups or complements of a noun, and an also optional sequence of relative clauses.

Examples:

- ... os livros cujo livreiro e' Freeman...
  (...the books whose publisher is Freeman...)

- ... o livro que foi escrito pelo ...
  (...the book that was written by Nilsson...)

- ... os artigos classificados em "ai applications"...
  (...the books classified under "ai applications"...)

A2) A verb phrase consists of an otional negation, a verb and its complements.

Example:

- ... nao e' um livro de Nilsson.
  (...is not a book by Nilsson.)

B) Articles

The following words are considered as articles, besides the classical ones (see subsection 4.4.6 for further discussion),

alguem	(somebody)
cada	(each)
nenhum	(none)
nenhuma	(")
nenhuns	(")
nenhumas	(")
ninguem	(nobody)
qualquer	(any)
quaisquer	(")
todo o	(every one)
toda a	(")
todos os	(every)
todas as	(every)

C) Nouns, verbs, adjectives and their complements

C1) There are two sorts of nouns: common nouns and proper nouns.

Examples:

common nouns:	artigo	(paper)
	livro	(book)
	publicacao	(document)
(single) proper nouns:		Bundy
		Hewitt
		Kowalski

Compound proper nouns:

"Computational Semantics"
"Logic for problem solving"
"Learning and executing generalized robot plans"

Compound proper nouns are recognized by the program on account of the quotation marks.

C2) The notion of adjective is flexible; the past principle of intransitive verbs is identified as an adjective, taking into account its function: to qualify the referred noun.

Examples:

escrito	(written)
ligado	(linked)
relacionado	(related)

C3) Verbs are defined through simple forms, in the present and in the past (singular and plural).

Examples:

e´	(is)
foi	(was)
sao	(are)

C4) All the complements are treated in the same way, and may have, for example, one of the following forms:

- ... os livros de Hewitt...	(the books by Hewitt)
- ... os livros escritos por Hewitt...	(the books written by Hewitt)
- ... tem como autor Winston...	(has as author Winston)
- ... tem Winston como autor...	(has Winston as author)
- ... um livro do Nilsson que foi publicado em 1971.	(a book by Nilsson that was published in 1971.)

C5) In the definition of each common noun, verb or adjective there is information about the kind of complements expected, with an indication about each optional introductory preposition.

For example, the word ´publicados´ may appear on its own or with a preposition (´em´ (in), ´apos´ (after), ´entre´ (between), etc.):

- ... os livros publicados...	(the books published)
- ... os livros publicados em...	(the books published in)
- ... os livros publicados apos...	(the books published after)
- ... os livros publicados entre...	(the books published between)

C6) Verb complements are not classified under the classical categories, such as direct or indirect object. However, there is a type of complement which is exclusive of the verbs 'ser' (to be) and 'estar' (to be): the subject complement.

C7) The subject complement may have one of the following two forms (underlined in the following examples):

C71) noun phrase

Example:

- ... e' o artigo que foi publicado... (is the paper that was published)

C72) adjective group

- ... os artigos sao classificados (the papers are classified)

D) Relative clauses

We only consider relative clauses of the restrictive type. They are introduced by relative pronouns and may be preceded by another phrase-- the restricted noun phrase. The relative clause operates as a definer.

Example:

- ... os artigos cujo autor e' Warren estao classificados?
(...are the papers whose author is Warren classified?)

4.4.4. DICTIONARY

The dictionary is defined by two main parts: the domain-independent dictionary or core vocabulary (in module GRAMMA of program TUGA) and the domain-dependent dictionary or peripheral vocabulary (module DICTIO of program TUGA). The composition of each part takes into account the semantics of the words considered.

The core vocabulary lists those words which are independent of any problem domain, such as articles, pronouns, adverbs, prepositions, conjunctions and numerals.

The peripheral vocabulary lists those words which depend on the problem domain, such as nouns, verbs and adjectives.

The semantics involves the description of the form, function and meaning of the basic syntactical categories.

--- CORE VOCABULARY

The domain independent dictionary covers the following words:

Definite articles

a (s)	(the)
o (s)	(the)

Indefinite articles and pronouns

alguem	(somebody)
algum	(some)
alguma	(some)
alguns	(some)
algumas	(some)
cada	(each)
nenhum	(none)
nenhuma	(none)
ninguem	(nobody)
qualquer	(any)
quaisquer	(any)
todo o	(every one)
todo a	(every one)
todos os	(every)
todos as	(every)
um	(a)
uma	(a)
uns	(a)
umas	(a)

Interrogative pronouns

quantos	(how many)
quantas	(how many)

Personal pronouns

me	(me)

Relative and Interrogative pronouns

onde	(where)
qual	(which; what; who)
quais	(which; what; who)
que	(what; that; which)
o que	(what)

quem	(who;whom)
cujo	(whose)
cuja	(whose)
cujas	(whose)
cujos	(whose)
o qual	(which)
a qual	(which)
os quais	(which)
as quais	(which)

Interrogative pronouns

quanto	(how much)

Adverbs

bem	(right)
como	(how)
mais	(more)

Prepositions

a	(to)
apos	(after)
ate´	(till)
com	(with)
de	(of;from)
desde	(since)
em	(in)
entre	(between;among)
para	(for)
por	(by)
sobre	(about)

Conjunctions

e	(and)
quando	(when)

Contractions of prepositions

`a	(to the)

ao	(to the)
aos	(")
do	(from the)
da	(")
dos	(")
das	(")
no	(in the)
na	(")
nos	(")
nas	(")
num	(in a)
numa	(")
nuns	(")
numas	(")
pelo	(by the)
pela	(")
pelos	(")
pelas	(")

Numerals

um	(one)
dois	(two)
tres	(three)
quatro	(four)
cinco	(five)
seis	(six)
sete	(seven)
oito	(eight)
nove	(nine)
dez	(ten)

--- PERIPHERAL VOCABULARY

The domain dependent dictionary covers the following words:

Common nouns

ano(s)	(year(s))

artigo(s)	(paper(s))
autor(es)	(author(s))
bibliografia	(bibliography)
classe	(class)
classificacao	(classification)
classificacoes	(classifications)
conversa	(conversation)
data(s)	(date(s))
dialogo	(dialogue)
documento(s)	(document(s))
editor(es)	(editor(s))
escritor(es)	(writer(s))
especie(s)	(variety(s))
feminina	(feminin)
feminino	(")
informacao	(information)
informacoes	(informations)
livreiro(s)	(publisher(s))
livro(s)	(book(s))
masculina	(masculin)
masculino	(")
nome	(name)
pessoa(s)	(person(s))
publicacao	(publication)
publicacoes	(publications)
referencia	(reference)
relatorio(s)	(report(s))
tipo(s)	(type(s);man)
titulo(s)	(title(s))

Note: The list of proper nouns is excluded, and it can be consulted in the program, namely in its module DICTIO.

Verbs

apagar	(to delete)
apague	(delete)
arquivar	(to store)
arquive	(store)

atribui	(assigns)
chama	(calls)
classifica	(classifies)
classificar	(to classify)
classifique	(classify)
concordo	(I agree)
conhece	(know)
criar	(to generate)
crie	(generate)
deixe	(leave it)
depende	(depends)
dependem	(depend)
desisto	(I give up)
destruir	(to destroy)
diz	(says)
dominar	(to dominate)
domina	(dominates)
e´	(is)
escreve	(writes)
escrevem	(write)
escreveram	(wrote)
escreveu	(wrote)
esquecer	(to forget)
esta´	(is)
estao	(are)
existe	(exists)
existem	(exist)
foi	(was)
ha´	(there is)
mude	(change it)
publica	(publishs)
publicam	(publish)
publicaram	(published)
publicou	(published)
quer	(want)
sao	(are)
ser	(to be)
significa	(means)
significam	(mean)
tem	(has)
tenho	(have)
ter	(to have)

Adjectives etc:

abaixo	(below)
acima	(above)
arquivada(s)	(stored)
arquivado(s)	(stored)
autonoma	(self contained)
autonomo	(self contained)
chamado	(called)
classificada(s)	(classified)
classificado(s)	(classified)
conhecido	(known)
dependente(s)	(dependent)
dominante(s)	(dominated)
editada(s)	(edited)
editado(s)	(edited)
escrito(s)	(written)
junto	(near)
ligada(s)	(connected)
ligado(s)	(connected)
publicada(s)	(published)
publicado(s)	(published)
referenciada(s)	(referenced)
referenciado(s)	(referenced)
relacionada(s)	(related)
relacionado(s)	(related)

4.4.5. LEXICAL SYNTAX AND SEMANTICS

In this subsection we are only concerned with the meaning of such words belonging to some basic syntactical categories, as nouns, verbs and adjectives, and with their relationship to other words of the language in a certain world. Some other aspects of semantics such as the description of how the meaning of a string of words

in the language is made up of the meanings of the individual words in the string, are discussed in subsection 4.4.6.

The meaning of words like nouns, verbs and adjectives is described by specifying the property they introduce and by characterizing the associated arguments. A property is a data base predicate rule providing a way to refer to the closed library world. An argument is an individual or a set of individuals.

With each argument there is associated the following parameters: the domain (D), the gender (G), the number (N), and optionally the case (K).

The domain D is a member of the hierarchy of domains.

Example: D=aut([]) means D is a terminal domain called 'aut' (authors)

The gender G may take two values: 'mas' (masculine) and 'fem' (feminine). There is no neuter in Portuguese.

The number N may take two values: 'sin' (singular) and 'plu' (plural).

The gender and the number define the agreement (A).

A=G-N

(the functional symbol '-' connects the parameters G and N;)

The case K may take the following values: 'sub' (subject/nominative), 'dir' (direct object/accusative), 'noun' (subject complement) and 'pre' (preposition);

The definitions for common nouns ('noun'), and proper nouns ('pn'), verbs ('verb'), and adjectives ('adj'), present the following general forms:

```
noun([A-D-X,...L],pr(property)) --> no(noun,A);
pn(proper noun,gender,domain);
verb([A-D-X,...L],pr(property)) --> ve(verb,A);
adj([A-D-X,...L],pr(property)) --> ad(adjective,A);
no(Type,G-N)--> [Name], {nol(Name,Type,G-N)};
nol(noun,noun group,gender-number);
ve(Type,N)--> [Verb], {vel(Verb,Type,N)};
vel(verb,verb group,number);
ad(Type,GN)--> [Adj], {adl(Adj,Type,GN)};
adl(adjective,adjective group,gender-number);
```

where X is the variable associated with the first argument of the property; L is a list of items of the form K-A-D-X, one for each other argument of the property: if the property has only one argument then L is the empty list. Properties introduced by nouns, verbs and adjectives are embedded in the unary term 'pr'. This term handles the manipulation of each property as regards

retrieval and relationship of their arguments, considered as individuals or sets of individuals.

We observe that these grammar rules are indexed by the input words, which are stored as unit clauses. This fact allows the enlargement of the dictionary during the dialogue.

Examples:

(1) Definition of the common noun 'escritor' (writer);

```
noun([A-typ(aut([]))-X],pr(author(X))) --> no(aut,A);

nol(escritor,aut,mas-sin);
```

(2) Definition of the proper noun 'Winograd'

```
pn(winograd,mas,typ(V));
```

(3) Definition of the verb 'publicar em' (to be published by);

```
verb([(G-N)-typ(aut([]))-X,prep(em)-_-typ(pub([]))
                -Y],pr(pub_of_aut(Y,X))) --> ve(pub,N);

vel(publica,pub,sin);
vel(publicou,pub,sin);
vel(publicam,pub,plu);
vel(publicaram,pub,plu);
```

(4) Definition of the adjective 'publicado entre' (published between);

```
adj([A-tit(V)-X,prep(e)-_-yea([])-Z,
                prep(entre)-_-yea([])-Y],
    pr(published_between(X,Y,Z))) --> ad(pub,A);

adl(publicado,pub,mas-sin);
adl(publicada,pub,fem-sin);
adl(publicados,pub,mas-plu);
adl(publicadas,pub,fem-plu);
```

Observe the ordering of the list of arguments. The second and the third members are inverted on account of Colmerauer´s precedence rule: "whenever an adjective has two complements, the quantification is made in the inverse order of their natural order of appearance".

The relationship of a word with other words of the language is ruled by the definition of that word through two simple semantic agreement rules.

An agreement rule is a type of constraint on the form of words occurring together. There are two types of constraint: i) one regarding the domain of the words; ii) the other regarding the agreement (gender and number).

The specification of a domain for each argument (represented as a property´s variable) establishes the relationship of the word to be defined with other words of the language, and detects semantic anomalies. The domain specifies the universe of possible individuals, upon which are made restrictions defined by the relations.

Example: the program doesn´t accept the sentence,

Elsevier publicou na "North-Holland".
(Elsevier published at "North-Holland".)

because ´Elsevier´ is a publisher and not an author. The verb ´publicar em´ (to be published by), defined above, accepts only as values for variable ´X´ the

members of the domain of authors (D=typ(aut([]))).

The specification of the agreement, by gender or by number, also detects lack of gender agreement, such as that exhibited by the following example,

Example: the sentence,

As artigos classificadas...
(The classified papers...)

has not gender agreement because the article 'as' (the) and the adjective 'classificadas' (classified) are in feminine-plural and the common noun 'artigos' (papers) is in masculine-plural. The noun is the head of the sentence because its form governs either the form that the other two words may take or what kind of word or word sequence is possible in the rest of the sentence.

4.4.6. SYNTAX AND SEMANTICS

1. Our grammar of Portuguese establishes how syntactically correct Portuguese sentences are formed and what their meaning is.

The grammar (module GRAMMA of program TUGA) is a set of rules which formalise the structure of the Portuguese language, block the formation or acceptance of ungrammatical sentences, and build up a logical

description of meaning. Some of these regularities exist in other languages, and in particular, some are common to other Romance languages, such as Spanish (Dahl,1977), and French (Colmerauer,1977;Pique,1978).

2. A brief list of the main syntactical rules for the fragment of Portuguese is given below. It covers the form of user's sentences and phrases. For further details consult program TUGA in Appendix 3.

N.B. Non-terminals are shown without their arguments.

Main sentence

(1) prin --> pre_locl, prop, final.
(2) prin --> int_rell, prop, final.
(3) prin --> prop, final.

Phrase(s)

(4) prop --> nucleus, compls, continue.

(5) nucleus --> arg, neg, verb.
(6) nucleus --> arg, neg, ve, advg_g.
(7) nucleus --> verb, per_pron.
(8) nucleus --> verb.

Negation

(9) neg --> [nao], {negative}.
(10) neg --> [neg].
(11) neg --> [ ].

Subject placement at the beginning of the sentence

```
(12)  int_rel1 --> int_rel2.
(13)  int_rel1, [nucleus], [mov_arg] --> int_rel2, arg,
                                         nucleus.
```

Subject, interrogative article and relative pronoun generation

```
(14)  int_rel2, [arg] --> pron.
(15)  int_rel2, [case], [art] --> case, interrog_art.
(16)  int_rel2, [case], [s_nucleus], [mov_arg] --> case,
                              pron, def_mark, s_nucleus.
```

Complements

```
(17)  compls --> [mov_arg], compls.
(18)  compls --> [ ].
(19)  compls --> compls, arg.
```

Subject inversion and coordenative conjunctions

```
(20)  arg, [neg], [ve] --> neg, ve, inv_mark, arg.
(21)  arg, [neg], [verb] --> neg, verb, inv_mark, arg.
```

Arguments (subjects)

```
(22)  arg --> case, n_phrase.
```

Noun phrases

```
(23)  n_phrase --> proper_nouns.
(24)  n_phrase --> def_art, proper_nous.
(25)  n_phrase --> s_nucleus, compls, relatives.

(26)  s_nucleus --> art, adjs, noun, adj_g.
```

Relative phrases of restrictive type

```
(27)  relatives --> int_rell, prop, relatives.
(28)  relatives --> [ ].
```

Disjunction or Conjunction of proper nouns

```
(29)  proper_nouns --> pns.

(30)  pns --> def_art, pns.
(31)  pns --> [P], {pn}.
(32)  pns --> [ ].
```

Adjectives and adjectival groups

```
(33)  adjs --> [ ].
(34)  adjs --> adj, compls, adj_g.

(35)  adj_g --> [ ].
(36)  adj_g --> adj, compls, adj_g.
```

Adverbial groups

```
(37)  advg_g --> adv_g.
(38)  advg_g [compls] --> compls, adv_g.
(39)  advg_g --> adver, s_nucleus, compls.
```

Ellipsis and generation of an article according the case

```
(40)  inv_mark, [case(_), none] --> none.
(41)  inv_mark --> [ ].

(42)  def_mark, [def_art] --> [ ].

(43)  case(K) --> [case(K)].

(44)  case(sub), [none] --> none.
(45)  case(sub), [not_all] --> not_all.
(46)  case(sub) --> [ ].

(47)  case(dir), [ind_art] --> [ ].
(48)  case(dir) --> [ ].

(49)  case(noun), [art] --> [ ].
(50)  case(noun), [art] --> ind_art.
(51)  case(noun) --> [ ].

(52)  case(prep(P)) --> contract, [P], {prep(P)}.
```

Interrogative and relative pronouns

```
(53)  pron --> int_prons.
(54)  pron --> int_pron.
(55)  pron --> int_art.
(56)  pron --> rel_pron.

(57)  int_prons, [ve] --> (pron3_r_i ;  pron1_r_1), ve.
(58)  int_prons, [ve] --> pron3_r_i ;  pronq_r_i.
```

3. The fragment of the syntax considered (set of 58 rules), shows only some essential aspects of the user's discourse. The following discussion refers to this fragment, by noting the numbers of the rules referred to, and by explaining in greater detail other parts of the grammar not covered by this brief fragment, such as articles and special verbs.

The rules are of formation type. The formation rules characterize only one structure (tree diagram), with only one meaning at a time, and operate on single symbols. Some of these rules are more complex, and we call them complex formation rules. They operate upon more complex objects, and perform functions similar to those played by transformational rules in a transformational grammar.

Complex formation rules are of four types:

i) deletion rules--delete some element from the input

ii) substitution rules--substitute an element for some other element

iii) insertion rules--insert an additional element

iv) movement rules--switch the order of elements

In the following, we explain the grammar rules by covering syntactic and semantic aspects concerning the main sentence, phrases, open questions and relative clauses, articles and special verbs.

I) Main sentence

The formation rules 1, 2 & 3, 'prin', state that a sentence may be initiated by a pre-locutory expression, 'pre_locl', such as "Tell me if", by a special grammatical class of interrogative or relative words, 'int_rell', or by the main phrase, 'prop'. These three cases correspond to questions not expressed in the interrogative form, open questions, and closed questions, statements or commands, respectively.

Each type of sentence is labelled by a token which appears in its logical structure. A clean separation among user types of request facilitates their further treatment (eg. answering, remembering, etc.).

The available tokens are:

question(conj) - for the conjunction of two questions,

question - for questions not expressed in the interrogative form, and for closed questions,

howmany - for open questions starting with an interrogative article 'quantos' (how many), optionally preceded by a preposition, and by an interrogative pronoun followed by verb 'ser' (to be) or 'estar' (to be);
these two linguistic constructions have different skeletons;

howmany([G-X,D],O) (1)

howmany(G-D-X,O) (2)

respectively, where 'O' represents the logical structure of the question; 'G' and 'D' are the gender and the domain associated to variable 'X'; 'X' represents the unknown quantity requested by the question: an individual in (1), and a set in (2).

which - for open questions starting with interrogative pronouns 'qual' (which; what) and 'quem' (who) followed by verbs 'ser' or 'estar' (to be), and by any other interrogative pronoun and verb;
these two linguistic constructions have different skeletons;

which((G-N)-D-X-O)

which((G-N)-D-X,K,O)

respectively, where 'O' represents the logical structure of the question; 'G', 'N', 'D' and 'K' are the gender, number, domain and case associated with the variable 'X', the unknown quantity requested by the question.

fact(conj) - for the conjunction of two statements;

fact - for single statements; and

order - for commands.

A sentence is always formed by a phrase, 'prop', and an end, 'final'. The end is responsible for detecting the final punctuation mark of the sentence and for

ascribing the corresponding token to the sentence's structure:

II) Phrase(s)

Formation rules 4, 5, 6, 7 & 8 state that a phrase is composed of two parts: nucleus, 'nucleus', and complements, 'compls'. A phrase may also be followed by another phrase, separated by a comma and/or a conjunction 'e' (and), and this is recognized by 'continue';

The phrase nucleus is composed by the subject (argument), 'arg', optionally followed by negation, 'neg', and a verb, 'verb'. However other forms are available for the nucleus. Rule 6 defines a particular verb phrase with adverbial group which appears in the following sentences;

"Artificial Intelligence" tem como autor Nilsson. (1)
("Artificial Intelligence" has as its author Nilsson.)

"Artificial Intelligence" tem Nilsson como autor. (2)
("Artificial Intelligence" has Nilsson as author.)

Rules 7 & 8 define the nucleus of typical imperative statements, as for example the following sentences,

De-me a classificacao de "Artificial Intelligence"! (3)
(Give me the classification of "Artificial Intelligence"!)

Classifique a "Artificial Intelligence"! (4)
(Classify "Artificial Intelligence"!)

III) Open questions and relative clauses

Formation rule 2 handles the analysis of open questions through the machinery behind rules 12 & 13 for interrogative or relative pronouns 'int_rell'. The same machinery also applies to relative clauses, through rules 27 & 28.

The relative clause bears a large number of similarities to open questions; hence it is possible to rely on a number of conclusions about open questions without an excessive amount of repetition in the analysis of relative clauses, since the relative pronouns appear, in general, to be the same as the interrogative pronouns (relative pronoun 'whose' is an exception). Relative clauses systematically lack exactly one noun phrase in a position where we would normally expect such a noun phrase to appear on the basis of the verb. Furthermore the interrogative pronoun always appears in initial position, and there is always a noun phrase missing elsewhere in the sentence, except when the interrogative pronoun is the subject of the sentence. The type of each phrase is ascribed in one argument of 'int_rell'.

We consider first open questions, and secondly relative clauses.

Open questions involve movement rules able to switch words or phrases from one position to another within the sentence. They are characterized by the interaction between argument inversion and the presence of a wh-word (interrogative pronoun or article) in sentence initial position.

Example: The declarative sentence;

Nilsson publica na "McGraw Hill". (5)
(Nilsson is published by "McGraw Hill".)

has the same propositional content as the interrogative sentence;

Onde publica Nilsson? (6)
(By whom is Nilsson published?)

characterized by two movements:

a) the complement 'na "McGraw Hill"' (by "McGraw Hill"') was moved to the beginning of the sentence and masked under the interrogative pronoun 'onde' (where);

b) the subject 'Nilsson' was moved to after the verb.

This example suggests one of the three rules (17, 18 & 19) defining the complements, namely rule 17: one of the complements, which appear after the verb in interrogative sentences, is moved to the beginning of a declarative sentence, to function as an interrogative pronoun. This kind of complement is called moving complement, 'mov_arg', and the rules responsible for its switch are the formation rule 12 and the complex formation rule 13.

Rule 12 states that the argument generated by the interrogative pronoun is at the beginning of the sentence. This is the case for sentences questioning about its argument, as

Quem e´ o author de "Artificial Intelligence"? (7)
(Who is the author of "Artificial Intelligence"?)

Rule 12 also states that ´int_rell´ is substituted by ´int_rel2´, in order to mark this occurrence.

Rule 13 switches the place of the moving argument, from after the nucleus to before the nucleus, renames it as argument ´arg´, and subtitutes ´int_rell´ by ´int_rel2´. Finally, the argument ´arg´ is transformed into the corresponding pronoun through complex formation rule 14.

In what concerns the movement of the subject ´Nilsson´ in sentence (6), a complex formation rule is required for dealing with the argument of the nucleus, ´publica Nilsson´ (publishes Nilsson). Such a rule belongs to a set of three rules for handling the subjects, rules 20 & 21 for the inversion of arguments and rule 22 for the standard construction of declarative sentences.

Complex formation rule 21 is able to recognize the switch of the argument from before the verb to after the verb. The rule ´inv_mark´ sets off an indication of that

inversion, used when a sentence starts with 'nenhum' (no) or with 'ninguem' (nobody). Rules 20 and 21 serve also to handle the argument inversion occurring in closed question, such as:

E' Nilsson um autor? (8)
(Is Nilsson an author?)

In the following figure we present the analysis of sentence (6):

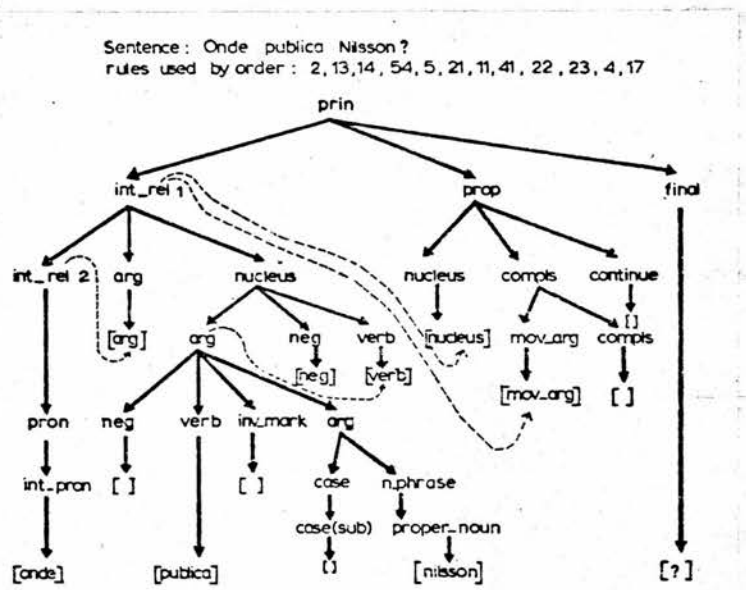


Fig. 23

Linguistic diagram

The diagram is a tree, a phrase structure for the sentence, constructed by the program during its search (failed branches are not considered). This tree is composed of two kinds of lines: solid lines represent expansions of non terminals, and broken lines represent the terminals inserted in the input string by the re-writing rules which have these terminals on their

left-hand side. At the bottom of the diagram are shown the constituents of the surface structure of the sentence, corresponding to the terminals of the whole tree structure. The order of use of the rules is also marked.

The diagram for declarative sentence (5), corresponding to the interrogative sentence considered previously, is presented in the next figure, where we observe the presence of one complex formation rule, 'contract', for governing the contraction of prepositions with articles.

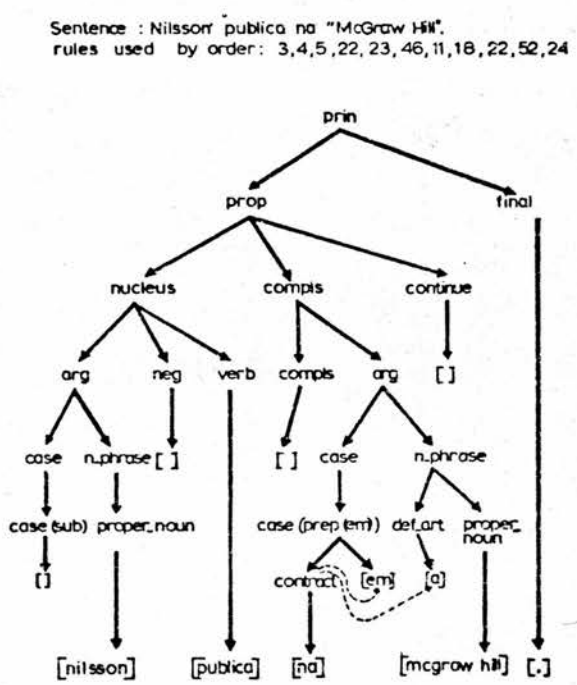


Fig. 24

Linguistic diagram

The open question;

Quantos autores nao escrevem na "North-Holland"? (9)
(How many authors do not write for "North-Holland"?)

whose diagram is presented in the following figure, shows the way the interrogative article is brought out by substitution rule 14, and the most general form of an argument, as expressed by rules 22, 23, 24, 25, & 26. On the other hand, it is also shown how rule 15 substitutes the article of the noun phrase 'n_phrase', by the interrogative article, 'interrog_art'.

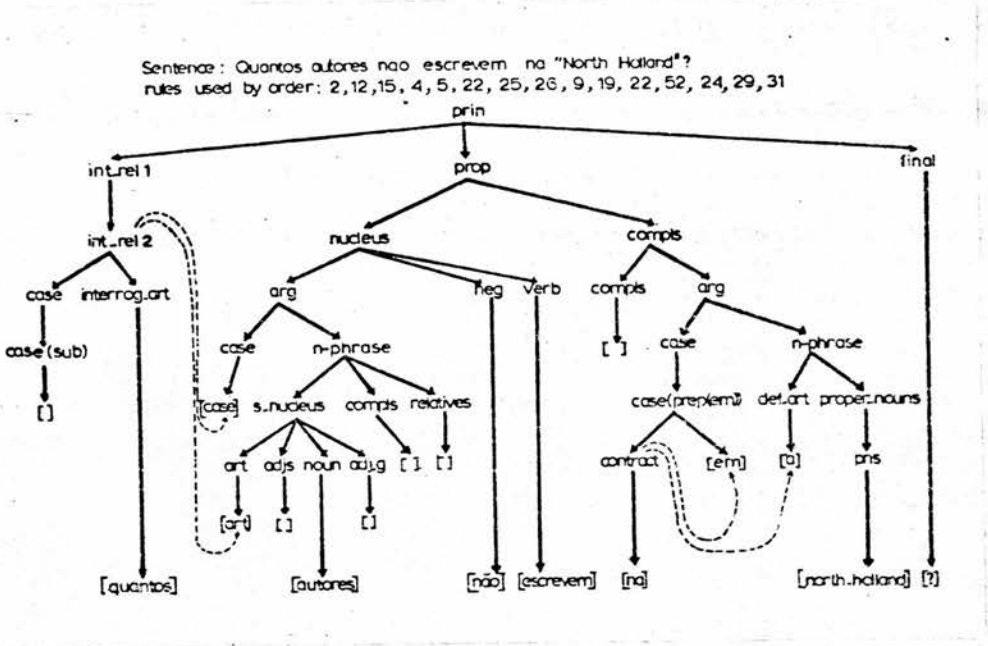


Fig. 25

Linguistic diagram

Let us now discuss how a relative clause is analysed. In its function, a relative clause, like an adjective, is a modifier of the noun. In many cases, it is possible to paraphrase a sentence that has an

adjective with one that has a relative clause.

Hewitt e´ um conhecido autor. (10)
(Hewitt is a known author.)

Hewitt e´ um autor que e´ conhecido. (11)
(Hewitt is an author who is known.)

The relative clause contains an argument, a verb and its complements, and is a constituent of a noun phrase. This fact allows the expansion of a noun phrase by attaching a relative clause. This attachment may be applied indefinitely, just as long as there is a noun phrase. This phenomenon, called recursion, is a property of natural languages.

Formation rules 25, 27 and 28 show the syntactical construction of relative clauses, and the process of recursion. Rule 27 introduces 'int_rell', which recognizes the relative pronoun and generates the argument of the relative clause.

The analysis of sentence (11), is shown in the following diagram:

Sentence: Hewitt é um autor que é conhecido.
rules used by order: 3,4,5,22,46,11,19,18,22,50,25,26,18,27,12,14,56,4,5,18,28

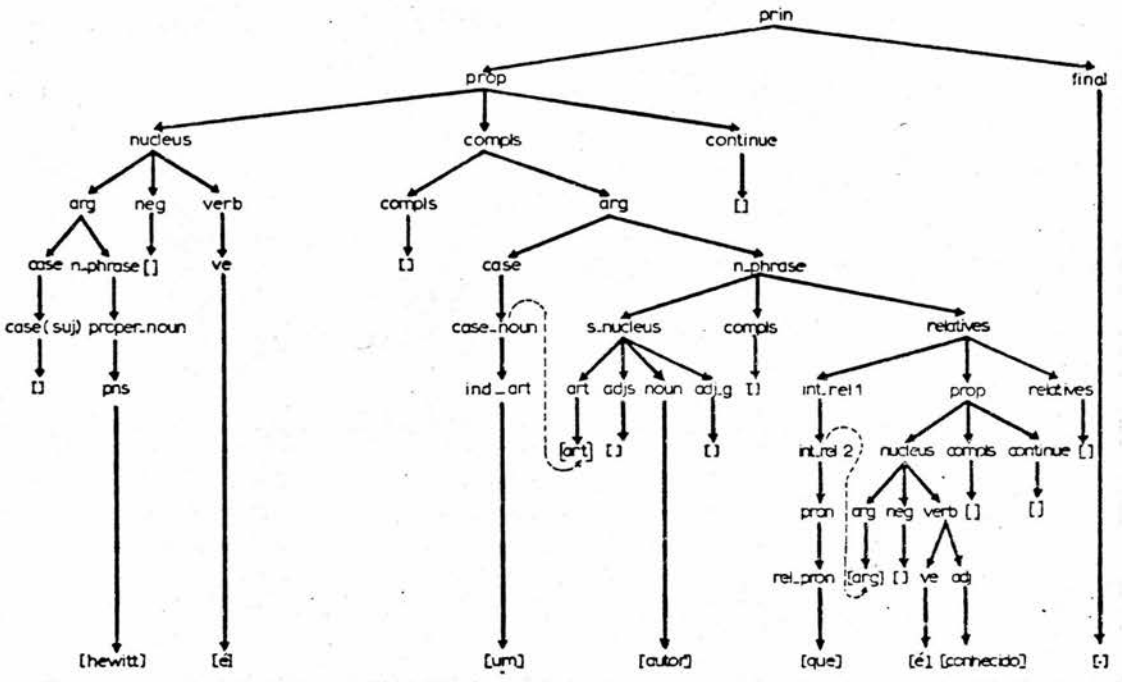


Fig.26

Linguistic diagram

Another open question:

Quais os autores cujo livreiro e´ Elsevier? (12)
(Who are the authors whose publisher is Elsevier?)

introduces again the machinery 'int_rel1' used either for open questions or for relative clauses.

Complex formation rules 14 and 27 handle the interrogative pronoun and the relative pronoun, respectively.

Rule 14 was explained in a previous example. As regards relative clauses, formation rules 27 and 28 are responsible for their analysis. Rule 27 uses the machinery behind 'int_rell', and in particular rule 16 which takes care of the relative pronoun.

Let us explain substitution rule 16 with the relative clause

cujo livreiro e' Elsevier (13)
(whose publisher is Elsevier)

of sentence (12). We write it as

o livreiro dos autores e' Elsevier (14)
(the publisher of the authors is Elsevier)

to make explicit the role played by the relative pronoun.

The relative clause (13), like an adjective, is a modifier of noun 'autores' (authors) in sentence (12). The modifier is not simply a relative pronoun followed by a sentence. By comparing phrases (13) and (14) we note a movement transformation. The argument 'autores' (authors), playing as complement of 'livreiro' (publisher) in (14), is deleted, substituted for the relative pronoun 'cujo'(whose), placed as the head of the relative clause. Moreover, it is necessary to delete the definite article which is embedded in the pronoun. Rules 16 and 42 achieve both objectives: rule 16 substitutes the moving argument, 'mov_arg', by the pronoun, and rule 42 inserts the definite article taken out by 's_nucleus',

defined by rule 26.

The above open question (12) presents an ellipsis of the verb tense 'sao' (are), very common in this type of Portuguese questions. The ellipsis is handled by insertion rule 58, responsible for detecting the interrogative pronoun and for inserting the missing verb tense 'sao' (are). The following figure shows the analysis of this question.

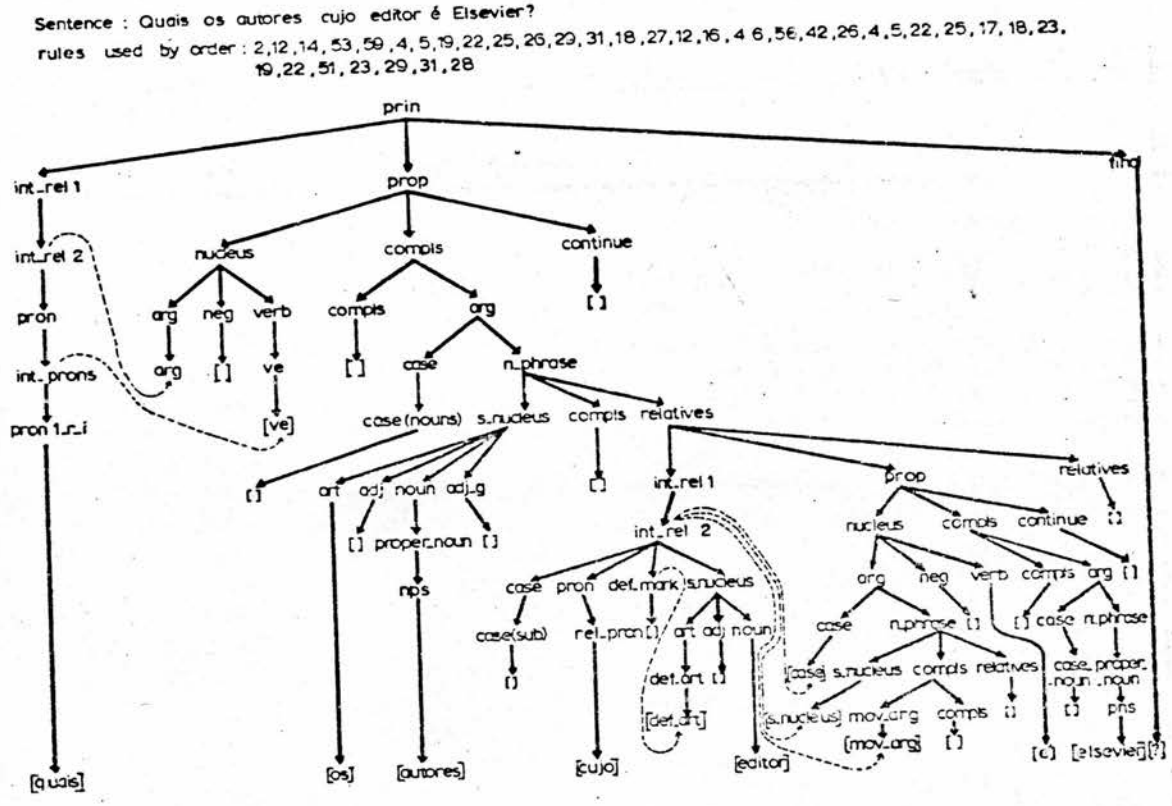


Fig.27

Linguistic diagram

IV) Articles

Articles are defined in Appendix 1 and discussed in Chapter 3. In the following, we consider the implementation of those definitions.

The semantics of an article is specified by a grammar rule, such as

```
art(A-D-X,O1,O2,for([X,D],O3,O4)) --> article.
```

In the left-hand side of the rule variable 'X' stands for the phrase subject, where 'A' and 'D' are its agreement (gender and number) and domain. 'O1' and 'O2' correspond to the properties attached to the subject and predicate. 'O3' and 'O4' are the logical formulae of our logical system, containing 'O1' and 'O2'. 'O4' stands also for the definition of cardinality of the article.

The right-hand side of the rule consists of a non-terminal or a terminal, optionally followed by a sequence of extra conditions (Prolog procedure calls). This non-terminal simply makes the access to the terminal forms.

1) Article 'o' (the)

```
art((G-sin)-D-X,O1,O2,for([X,D],O1,
          if(card(X,equal,1),O2))) --> def_art(G-sin).

def_art(A) --> [def_art(A)].
```

```
def_art(fem-sin) --> [a];
def_art(mas-sin) --> [o];
```

Example: the sentence containing a definite noun phrase;

Nilsson e´ o autor de "Artificial Intelligence";
(Nilsson is the author of "Artificial Intelligence";)

has the following description:

```
fact(
for([X,typ(V)];
    pr(author(X,[artificial intelligence]));
    if(card(X,equal,1);
       pr(set_equal([nilsson],X)))))
```

which is represented by the following tree:

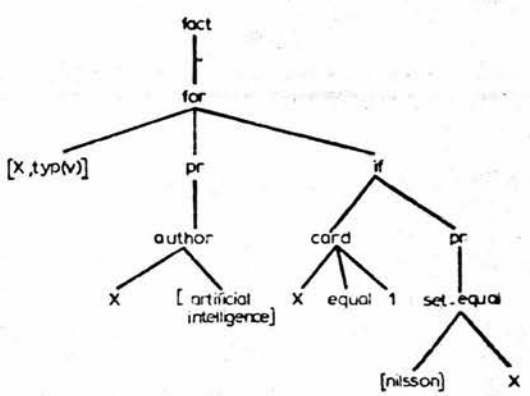


Fig.28

Logical tree structure

whose meaning is:

for those (sets of individuals) ´X´, belonging to the domain ´typ´ (types: authors and publishers), such that ´X´ is the author of "artificial intelligence", if the cardinality of ´X´ is equal to 1, then ´X´ is Nilsson;

2) Article 'os' (the)

```
art((G-plu)-D-X,O1,O2,for([X,D],O1,
        if(card(X,greater,0),O2))) --> def_art(G-plu);

def_art(fem-plu) --> [as];
def_art(mas-plu) --> [os];
```

3) Article 'um' (a)

```
art((G-sin)-D-X,O1,O2,for(([X,D],and(O1,O2),
        card(X,greater,0))) --> ind_art(G-sin);

ind_art(_), [pessoa] --> [alguem];
ind_art(mas-sin) --> [um] ;  [algum];
ind_art(fem-sin) --> [uma] ;  [alguma];
ind_art(A) --> [ind_art(A)];
```

This rule also serves for the interpretation of 'alguem' (somebody). As the meaning of this article is 'alguma pessoa' (some person), it is necessary to introduce the word 'pessoa' (person) as a left-hand terminal of rule 'ind_art'.

4) Article 'uns' (some)

```
art((G-plu)-D-X,O1,O2,for([X,D],and(O1,O2),
        card(X,greater,1))) --> ind_art(G-plu);

ind_art(mas-plu) --> [uns] ;  [alguns];
ind_art(fem-plu) --> [umas] ;  [alguns];
```

Rules 47, 48, 49, 50 and 51 support the interpretation of indefinite articles by handling their ellipsis and generation according to the case.

5) Article 'todo o' (every;all the)

```
art(A-D-X,O2,for([X,D],and(O1,not(O2)),
        card(X,equal,0))) --> todo_art(A).

todo_art(mas-sin) --> [todo,o].
todo_art(fem-sin) --> [toda,a].
todo_art(_-sin) --> [qualquer].
todo_art(mas-plu) --> [todos,os].
todo_art(fem-plu) --> [todas,as].
todo_art(_-plu) --> [quaisquer].
todo_art(_-sin) --> [cada].
```

In Portuguese the word 'todo' (every) appears on its own when it operates as an indefinite pronoun. In all other cases, the word 'todo' is always followed by the definite article. This rule serves also for the interpretation of 'cada' (each) and 'qualquer' (any). We observe that the Portuguese article 'qualquer' is similar to English article 'any'. It has two different senses ('every' and 'some') according to the context where is inserted.

Example:

> Qualquer autor escreve na North-Holland.
> (Every author writes for North-Holland.)
>
> A North-Holland nao tem qualquer escritor.
> (North-Holland doesn't have any writer.)

The impossibility of dealing with the two senses of 'any' is a limitation of Colmerauer's framework. The French article 'quelque' is not similar to 'any' because it has only one sense ('some').

6) Article 'nem todos os' (not all the)

```
art(A-D-X,O1,O2,for([X,D],and(O1,not(O2)),
     card(X,greater,0))) -->
             [not_all(A)], {negative}.

not_all(A) --> [nem], todo_art(A).
```

This rule operates with rule 45 and the definition of article 'todo_art'. The rule 45 states that 'not_all' imposes the case subject, and it calls the rule 'not_all'. In fact, this article is decomposed into the particle 'nem' (neither) and the article 'todos os' (all the).

The extra condition 'negative' handles negation constructions. It distinguishes standard constructions from double negation constructions, imposed by article 'nenhum' (no), as it is explained in the following.

7) Articles 'nenhum' (no) and 'nenhuns' (no)

```
art((G-sin)-D-X,O1,O2,not(for([X,D],and(O1,O2),
     card(X,greater,0)))) --> none(G-sin),{negative}.

art((G-plu)-D-X,O1,O2,not(for([X,D],and(O1,O2),
     card(X,greater,1)))) --> none(G-plu),{negative}.

none(mas-sin) --> [nenhum].
none(fem-sin) --> [nenhuma].
none(mas-plu) --> [nenhuns].
none(fem-plu) --> [nenhumas].
none(_), [pessoa] --> nonel.
none(_), [tipo] --> nonel.
none(A) --> [none(A)].

nonel --> [ninguem].
```

Article 'nenhum/nenhuns' (no) is supported by rules 40 and 44 for dealing with ellipsis and generation of an article. This rule also serves for the interpretation of 'ninguem' (nobody). As the meaning of this article is 'nenhuma pessoa' (no person), it is necessary to introduce the word 'pessoa' (person) as left-hand terminal of the rule 'none':

'nenhum' (no) imposes a double negation when it appears after the verb, and some auxiliary rules are needed for analysing the negation:

Let us consider two main constructions and the corresponding negation rules:

(1) Nilsson nao tem nenhum livro.
(Nilsson doesn't have any book.)

Surface structure: ...nao + verb + nenhum...
negation rule:

```
neg(O,O),[verb(X,O1),none(A)] -->
        [nao], verb(X,O1), none(A);
```

(2) Nilsson nao escreve para nenhum livreiro.
(Nilsson does not write for any publisher.)

Surface structure:

...nao + verb + prep + nenhum...

negation rule:

```
neg(O,O),[verb(X,O1),P,none(A)] --> [nao],
      verb(X,O1),[P],(prep(P)), none(A).
```

This kind of construction has a particular form:

Nilsson nao e´ autor de nenhum livro.
(Nilsson is not the author of any book.)

Surface structure:

...nao + aux verb + noun + prep + nenhum...

This construction is characterized by the appearance of a common noun between the auxiliary verb and the preposition. A new rule for copulative verb is needed:

```
verb([A-D-X,Y],O) --> copula(A), noun([A-D-X,Y],O).
```

because the rules for the complements cannot handle the kind of constructions generated by the negation rules.

The negation rules are completed by rules 9, 10 and 11. Rule 9 asserts the predicate ´negative´ when a negation occurs.

8) Numerals

```
art(A-D-X,O1,O2,for([X,D],and(O1,O2),card(X,equal,I)))
                                              --> nb(I)

nb(I) --> [I], {integer(I)}.
```

There is a particular case for numerals, shown in the following sentence:

Quais os 3 livros de Nilsson?
(What are the 3 books from Nilsson?)

The numeral 3 is preceded by a plural definite article 'os' (the). A new rule is therefore necessary for dealing with this kind of construction:

```
art((G-plu)-D-X,O1,O2,for([X,D],O1,
    if(card(X,equal,I),O2))) --> def_art(G-plu),nb(1).
```

V) Special verbs

We call "special" those verbs used in a particular fashion, some either as auxiliaries or as main verbs, and others without argument. Verbs such as 'ser' and 'estar' (to be), 'ter' (to have) and 'haver' (to exist) belong to this class.

V.1) Verbs 'ser'(to be) and 'estar'(to be)

'Ser' and 'estar' are main verbs normally used to denote existence of, or to give information about, a person or thing:

Elsevier e' editor.
(Elsevier is a publisher.)

This use is governed by the following rule,

```
verb([(G-N)-D-X,noun-A-D-Y,pr(set_equal(X,Y))) -->
                                        ve(ser,N).

ve(type,N) --> [Verb], {vel(Verb,Type,N)}.

vel(e', ser,sin).
vel(foi,ser,sin).
vel(sao,ser,plu).
vel(foram,ser,plu).
```

```
vel(esta´,estar,sin).
vel(estao,estar,plu).
```

The verb establishes an equality between the argument and the subject complement. This equality is ascribed to the verb´s definition through the inclusion of the property ´set_equal´, whose meaning is set equality.

´Ser´ and ´estar´ are also copulative or linking verbs. In this case, the verbs do not introduce any property.

The copula ´ser´ (to be) stipulates three possible kinds of relationship between two nouns A and B:

i) A and B refer to the same individual (A=B)

(both A and B are specified, and can occur in either order)

ii) A belongs to the class B (A ∈ B)

(only A is specified, where as B necessarily refers to any number of individuals)

iii) All members of A are members of B (A ⊂ B)

(neither A nor B refers to a specific individual).

Let us take an example of ii)

A Elsevier e´ uma livreira.
(Elsevier is a publisher.)

The subject complement ´livreira´ (publisher) introduces the case ´noun´, defined by rule 51. It may be constructed with an indefinite article as above, or

with a definite article.

The example shows that the subject complement introduces a property associated with the argument 'Elsevier', and a new rule is therefore necessary for taking into account this new fact;

```
verb([A-D-X,noun-A1-D1-X],t) --> copula(A).

copula(_-N) --> ve(ser,N).
copula(_-N) --> ve(estar,N).
```

where 't' represents a property which is always true. The case is governed by rules 49, 50 and 51. Rules 49 and 50 serve to capture the property introduced by the subject complement, either as a result of the lack or presence of the indefinite article. Rule 51 takes into account all the other forms.

'Ser' and 'estar' are also auxiliary verbs used with the past participle to form the present perfect and the past perfect:

> "Artificial Intelligence" está classificado.
> ("Artificial Intelligence" is classified.)

This use is governed by the following rule,

```
verb([A-X,..L],O) --> copula(A),adj([A-X,..L],O).
```

where the past participle is considered as an adjective.

V.2) Verb 'ter'(to have)

'ter' is a main verb meaning 'have' which requests a direct object;

"Computational Semantics" tem uma classificacao.
("Computational Semantics" has a classification.)

This use is governed by the following rule;

```
verb([(G-N)-D1-X,dir-A-D2-Y],pr(have(D1,X,D2,Y))) -->
                                        ve(ter,N);
ve(Type,N) --> [Verb], {vel(Verb,Type,N)};
vel(tem,ter,_);
```

The verb introduces the relation 'have' which depends on the argument and direct object's domains. So, with this double dependence we have only one verb rule for several uses of 'have', which arise on account of the interchange of its argument and its direct object. The verb introduces the case 'dir' which is defined by rules 47 and 48;

Here are examples of several uses of verb 'ter' (have):

Que livreiros tem Nilsson?
(What publishers has Nilsson?)

Quantos livreiros tem Nilsson?
(How many publishers has Nilsson?)

Nilsson tem um livreiro.
(Nilsson has one publisher.)

Quem tem livreiros?
(Who has publishers?)

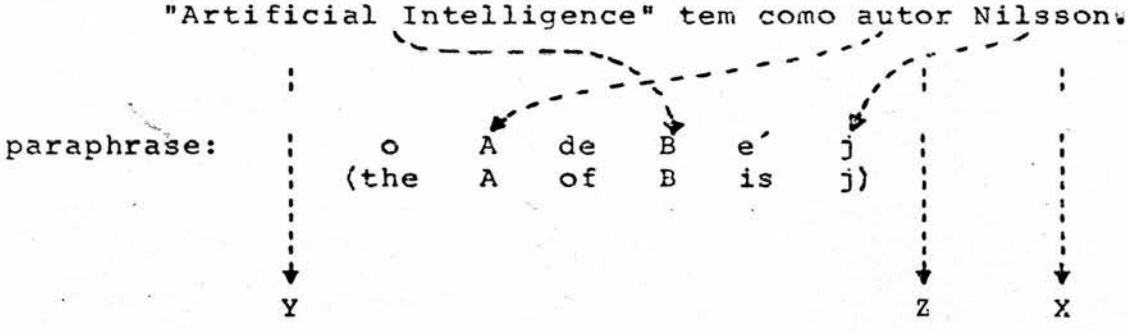
Verb 'ter' is also used in adverbial constructions, such as;

"Artificial Intelligence" tem como autor Nilsson.
("Artificial Intelligence" has as author Nilsson.)

which has an alternative construction in Portuguese;

"Artificial Intelligence" tem Nilsson como autor.
("Artificial Intelligence" has Nilsson as author.)

Rule 6 is able to deal with both constructions through the definition of adverbial groups and the introduction of the property 'set_equal'. Let us observe with more detail how this rule is built up;



("Artificial Intelligence" has as author Nilsson.)

This paraphrase is achieved by the following rule (rule 6 with explicit arguments);

```
nucleus([noun-A-D-Z],set_equal(X,Z),O2,O) -->
        arg(sub-Y,O4,O), neg(O3,O4), ve(ter,_),
        adv_g(A1-D-Z,prep(de)-Y,O2,O3).
```

where the adverbial group is defined as;

```
adv_g(X,Y,O1,O) --> adver, s_nucleus([X,Y,..L],v,O1,O2),
                    compls(L,O2,O).
```

These rules are written by comparing the ordering of the elements of the input string with the corresponding level in the current phrase structure tree. By doing so, we discover in what grammar rules it is necessary to do the right transformations. For the construction in discussion, a new rule was written upon rule 5, requiring the definition of adverbial group 'adv_g'.

V.3) Verb 'haver' (to exist)

'Haver' is a main verb used to denote existence of. It requires an argument,

Ha' um livro escrito por Charniak.
(There is a book written by Charniak.)

and introduces a case 'noun'.

This use is governed by the following rule,

```
verb([A-D-X,noun-A1-D1-X],t),[um,titulo] -->
                                    ve(have,sin).

vel(ha',haver,sin).
```

similar to the rule for verb 'ser' (to be) when the subject complement is preceded by an indefinite article, except for the introduction of the necessary argument. The required argument is introduced by the left-hand terminal 'um titulo' (a title).

4.5. ANALYSIS OF PORTUGUESE SENTENCES

The subset of Portuguese grammar, discussed so far, is written in Prolog as a DCG. It determines what to do with a Portuguese sentence, ie. it is simultaneously responsible for the syntactical analysis of a sentence and for its translation into a logical structure (see section 3.3). The syntactical analysis or parsing consists of showing that a string of words defines a sentence according to given rules of grammar. The translation consists of describing the sentence meaning as a program by plugging well-formed formulae into a logical structure.

We consider how TUGA achieves the analysis of Portuguese sentences by taking an example and displaying the overall analysis process on a tree. Three snapshots are presented to show its intermediate results.

Example: Let us consider TUGA grammar and
the following sentence,

O titulo que Nilsson tem e´ "Artificial Intelligence".
(The title that Nilsson has is "Artificial Intelligence".)

The analysis process is represented by a tree. We only consider the successful part of the search done by the grammar, and disregard all the backtracking. The execution machinery of Prolog is implicit.

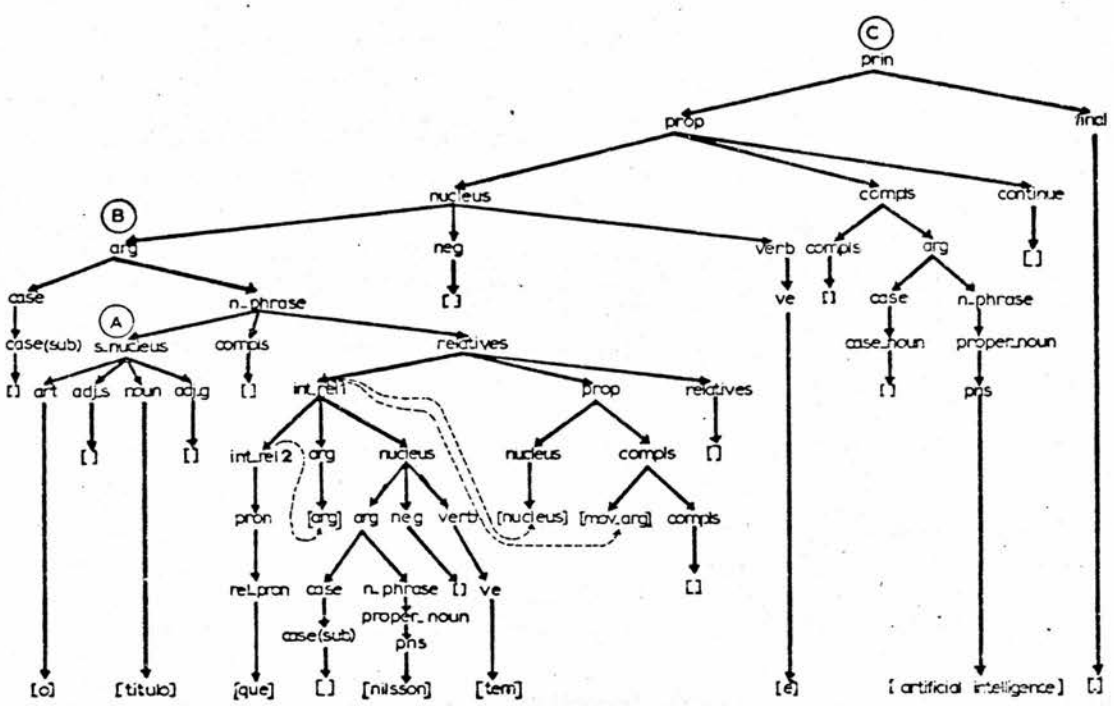


Fig. 29

Linguistic diagram

The analysis of the sentence, which was transformed into a list, is done from left-to-right. We choose three points A, B and C in the tree of the figure above to present fragment terms of the sentence analysed. Below each term is presented the sentence segment to be analysed further on.

Snapshot A

```
  s_nucleus([(mas-sin),tit(V),X],
            Y,
            pr(title(X)),
            Z,
            for([X,tit(V)],
                Y,
                if(card(X,equal,1),
                   Z)))

[que, nilsson,tem,´e´´´,artificial intelligence,.]
```

Snapshot B

```
  arg(sub-(mas-sin)-tit(V)-X,
      Z,
      for([X,typ(V)],
          and(pr(title(X)),
              pr(have(typ(aut(V)),[nilsson],tit(V),X))),
          if(card(X,equal,1),
             Z)))

[´e´´´, artificial intelligence,.]
```

Snapshot C

```
  prin(fact(
   for([X,tit(V)],
       and(pr(title(X)),
           pr(have(typ(aut(V)),[nilsson],tit(V),X))),
       if(card(X,equal,1),
          pr(set_equal(X,[artificial intelligence])))))))
[ ]
```

The snapshot C gives the final product of the DCG, the logical structure for the above sentence, recognized as an assertion. This structure is represented in figure 30.

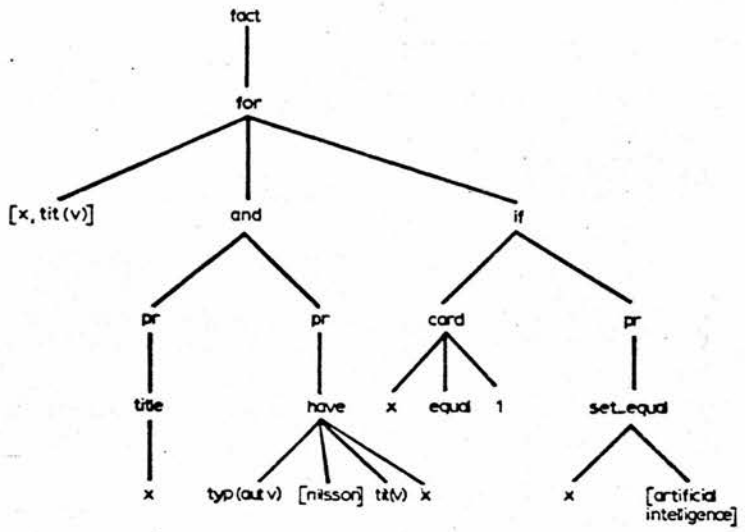


Fig. 30

Computational structure

The logical reading of this structure is:"for those 'X' belonging to the domain of titles such that 'X' is a title and 'nilsson' has 'X', it is true that 'X' equals [artificial intelligence] if the cardinality of 'X' equals 1".

4.6. COMPUTATION OF ANSWERS

Computation of answers consists of: 1) evaluating the logical structure corresponding to the translation of some sentence, and according to a certain logical system and to the items retrieved from the data base, and 2) generation of output messages.

In Chapter 3 we surveyed this matter by presenting a general description of the algorithm adopted for evaluating the structure. Now, we present implementation details by discussing several procedures through examples of different linguistic constructions.

If, in everyday dialogue, some question, statement or command is posed, we may decide on a certain answer, yet statements and command don't usually call for an answer. TUGA answers in all cases. The following dialogues are an example:

(1) u- Onde publica Nilsson?
(By whom is Nilsson published?)

p- Na "McGraw Hill".
(At "McGraw Hill".)

(2) u- Quem e' o livreiro de "Artificial Intelligence"?
(Who is the publisher of "Artificial Intelligence"?)

p- "McGraw Hill".

(3) u- E' Winograd o autor de "Programming in POP-2"?
(Is Winograd the author of "Programming in POP-2"?)

p- Nao.
(No.)

(4) p- Quer mais documentos?
(Do you want more documents?)

u- Nao quero. (No, I don't.

p- E que mais? And what else?)

(5) p- Por favor, de-me uma referencia do documento em questao? (Please, give me one reference of the document in question?)

u- Desisto. (I give up.)

p- Esta´ bem.
E que mais?

(All right.
And what else?)

Consider the piece of dialogue (1). The request posed is an open question, starting with the interrogative pronoun ´onde´ (where), and composed of an ordinary verb ´publicar´ (to publish) plus the preposition ´em´ (at). The interrogative pronoun ´onde´ (where) is a member of a set of wh-words which are analysed as:

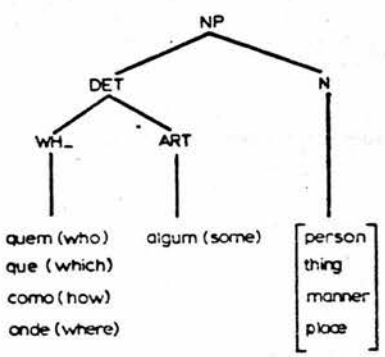


Fig.31

Set of wh-words

The declarative sentence, corresponding to the question,

Nilsson publica na "McGraw Hill".
(Nilsson is published by "McGraw Hill".)

shows explicitly a prepositional phrase and the quantification introduced by the definite article ´a´ (the), through the contraction ´na´ of preposition ´em´

(at) plus the definite article 'a' (the). However, the quantification is not explicit in the question. The answer of dialogue (1);

Na "McGraw Hill";
(At "McGraw Hill";)

contains a contraction, where the definite article 'a' agrees with the gender of the proper noun "McGraw Hill" (the gender of 'editora' (publisher) is feminine), which is the data item retrieved.

Consider now the previous dialogue piece (2). The question posed is an open one, starting with the interrogative pronoun 'quem' (who), and followed by verb 'ser' (to be). This verb requires a definite article, 'o' (the), and therefore the quantification is introduced explicitly. The question is built upon a noun phrase quantified by the definite article.

The answer,

"McGraw Hill".

is a proper noun, the data item retrieved.

Consider now the previous dialogue piece (3). The question posed is a closed one, and imposes a yes/no answer.

Pieces of dialogue (1), (2) and (3) refer to the program's library knowledge. However, the program is also able to accept user sentences not referring to that

knowledge or undoing the action that had already started.

Consider now dialogue piece (4). The program asks whether more data is required, and receives a negative answer.

Consider now the previous dialogue piece (5). The program interrogates the user during the classification process of a document. The user gives up, and the program stops the current dialogue and returns to the main dialogue course.

These examples show clearly that the program may access different sorts of knowledge in order to construct an appropriate answer: the syntactical and the semantic knowledge, and the logical system. The syntactical knowledge groups the aspects concerning the linguistic form of the sentence (eg. the interrogative marker and main verb for questions). The semantic knowledge concerns the logical interpretation of the sentence. The logical system supports the program's evaluation of that logical structure.

The program also needs computational knowledge for guiding and carrying out its answering function. The guidance is provided by the grammar module of the program. The grammar constructs a skeleton for each sentence's type, in which is embedded its logical structure and some syntactical information about the

sentence.

Example: the user sentence of dialogue (1),

Onde publica Nilsson?
(By whom is Nilsson published?)

has the following skeleton, where the third argument represents the logical structure of the sentence,

```
which((G-N)-typ(pub(V))-X,
      prep(em),
      pr(
         pub_of_aut(X,coord([nilsson]))))
```

where

'which' is the token for qualifying the question type,
'G' is the gender of 'X',
'N' is the number of 'X',
'typ(pub(V))' is the domain of 'X'
'X' is the variable standing for the requested data item(s);
'prep(em)' is the case,
'pr(P)' is a term to allow the handling of property 'P',
'P' is the property introduced by the verb 'publicar' (to publish), and it relates the sentence participant 'Nilsson' with the variable 'X'. The participant 'Nilsson' is inside a term marked by 'coord' which appears only for n-ary relations (n greater than 1). Here, the predicate 'coord' is of no use, because its argument is a one-member list, ie. there is no coordination in the sentence. Otherwise, it determines a certain kind of procedure for relating the arguments of the above 2-ary relation, and hence a certain data base search for selecting the possible individuals for 'X', one by one.

In figure 32, we present the classification of the kind of knowledge used to support the answering of user requests.

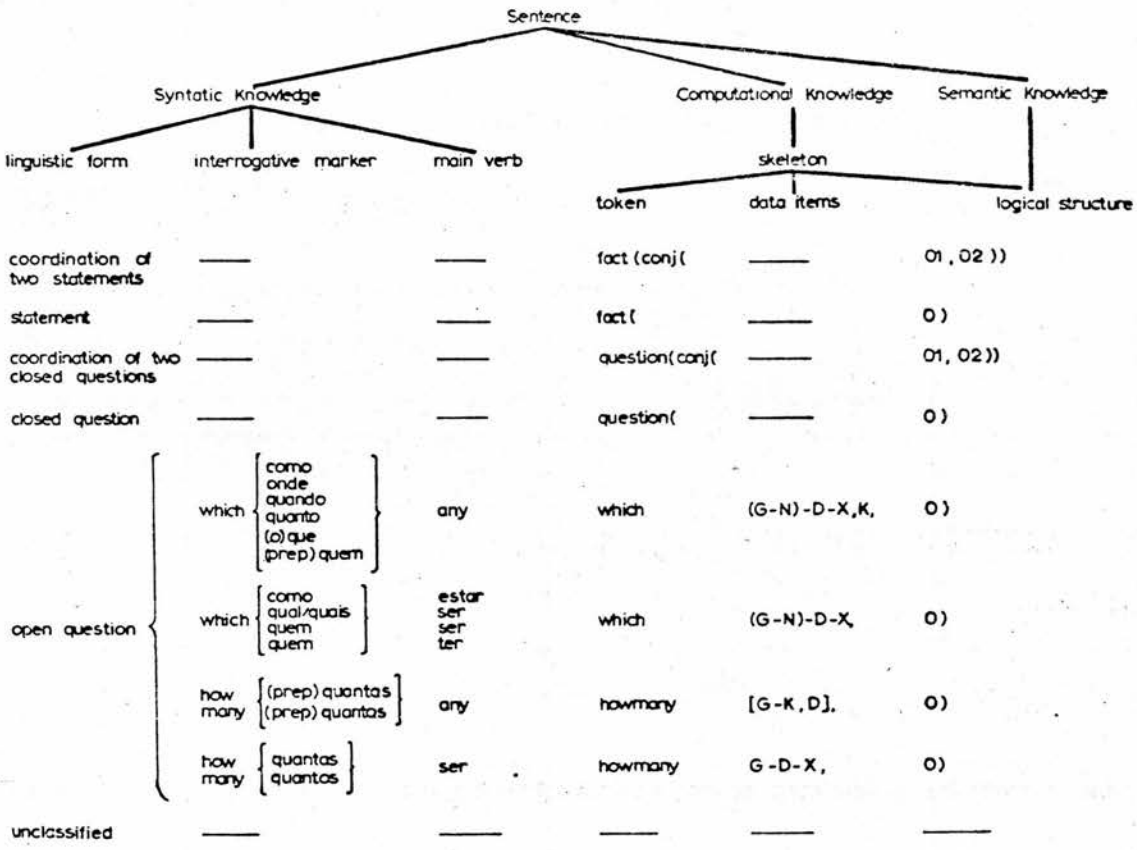


Fig:32

Knowledge for answering user requests

In brief, the computation of answers is the purpose of the TUGA semantic system, and it has three main aspects:

1) Accessing the data base to find or verify data items related to the user request. This is the retrieval aspect; it is dealt with by modules DBMS and DBASE of TUGA.

2) Evaluating the logical structure corresponding to the request sentence, on the basis of three truth-values: true, false and undefined. This is the evaluation aspect; it is dealt with by module DBMS of TUGA.

3) Forming the answer by choosing the appropriate standard form, and by inserting into it the data items selected. This is the output aspect; it is dealt with by module OUTPUT of TUGA.

These aspects are the subject of the following two subsections.

4.6.1. THE RETRIEVAL AND EVALUATION ASPECTS

The retrieval aspect is discussed by considering the treatment of singular versus plural, when quantification is present or presupposed, and the evaluation aspect by considering the treatment of conditional, of negation, and of meaningless.

We start by examining open questions because they present sufficient complexity to cover all the aspects dealt with by the other types of request.

Open questions are classified into two types: 'which' questions and 'howmany' questions. The second type deals with questions requesting a numeral, and the first one deals with all the other cases involving the retrieval of data items. Each type is divided into two subtypes: one for the ellipsis of an article (the quantification is presupposed), and the other for the explicit presence of articles co-occurring with special verbs. By so doing we separate different cases in the

specification of truth conditions. The four subtypes are all different as regards their respective skeletons. The subtypes regarding the ellipsis are similar for retrieval and evaluation purposes, and the same for the other two subtypes.

Retrieval and evaluation aspects are carried out by a general procedure called 'find_all' (see module DBMS in Appendix 3). This procedure finds all the individuals satisfying certain constraints, and eliminates the redundant ones. It requires a description of the individual(s) that is(are) the target of the search and an indication of the area to be searched. The description of the individual(s) is(are) available in the sentence's logical structure. The area of search is constrained by the domain of the individual(s) and its expected cardinality. The information on the cardinality of articles, present in their definition, allows the implementation of compatibility and incompatibility tests. These tests avoid unnecessary calculation of the cardinality and realize simple comparisons between the cardinality of the list of individuals in construction and the expected cardinality, each time a new individual is found. Compatibility tests specify a minimum of individuals, and incompatibility tests a maximum of individuals.

-- 'which' questions with ellipsis of an article

When the article is not present in the question, the logical structure is very simple. There is no quantification, no quantifier 'for(O1,O2,O3)', and thus no information regarding the cardinality (see the use of 'for' in the definitions of articles in Appendix 1). The information retrieval procedure 'find_all' presupposes quantification, and performs a large search for all individuals that satisfy the domain and the logical structure. The search is done through the domain and the n-ary relations introduced by verbs, nouns and adjectives. Individuals stored in the data base are accessed one by one.

Let us consider an example of a question of the type under discussion.

Example: the sentence,

Onde publica Nilsson? (1)
(By whom is Nilsson published?)

is paraphrased into,

if Nilsson has a publisher then who is he?

The sentence (1) has the following skeleton,

```
which([X,typ(pub(V))],
      prep(em),
      pr(pub_of_aut(X,coord([nilsson]))))
```

where the third argument is the logical structure.

The retrieval procedure 'find_all' verifies whether Nilsson has a publisher by examining, one by one, the individuals belonging to the area of publishers, defined by the domain 'typ(pub(V))', to see whether that individual represented by variable 'X', is a publisher of Nilsson. It does not consider the cardinality of 'X' because the quantification is not present in the sentence.

```
find_all([X,typ(pub(V))],
         pr(pub_of_aut(X,coord([nilsson]))),
         card(_,_,_),
         _)
```

Note that the fourth argument of 'find_all' corresponds to the truth value produced by the evaluation of the logical structure.

The computation of the logical structure consists of verifying the predicate embedded in the term 'pr', as many times as the existing possible individuals 'X' related with 'nilsson'. This verification is executed by imposing true as the expected truth value. The incompatibility and compatibility tests are of no use in this case because the cardinality is unknown. The term 'coord', inside the relation, imposes an operation of relating when its argument is a set of individuals. This is the case for the coordination of phrase elements, where the conjunction 'e' (and), linking proper nouns has the meaning of logical union, and not of logical

intersection, as it is currently considered in general cases.

The verification is not always so simple as in the example of sentence (1). In general, one has a term more complex than the simple 'pr' term discussed above. Therefore, it is necessary to decompose it into single patterns of verification, such as for 'for', 'and', 'non', and consequently for 'pr' and the inserted property 'P'. Each single verification applies to single individuals or to sets of individuals (see the discussion of Colmerauer's hypothesis in section 3.4).

A general verification consists either of a direct computation of the relation or of a relating operation. The direct computation of a relation serves to find the missing individual or to verify the relation validity. A truth value, true, false or undefined, is assigned to this computation, according to the result obtained. The relating operation serves to compare the sets of individuals, instantiated or to be instantiated in each relation argument. This comparison of successive individuals belonging to each argument of the relation covers four cases: i) the comparison of only one individual of a set to a single individual; ii) the comparison of each individual of a set to each individual of the other set; iii) the comparison of at least one individual of a set to a single individual; iv) the

comparison of all individuals of one set to all individuals of the other set. A truth value, true or false, is assigned to the relating procedure, according to whether the relation holds or does not hold. The truth value undefined is assigned when at least one argument of the relation to be verified is the empty list.

The execution of retrieval procedure 'find_all' involves the call of several procedures (see module DBMS of program TUGA), as it is shown in figure 33.

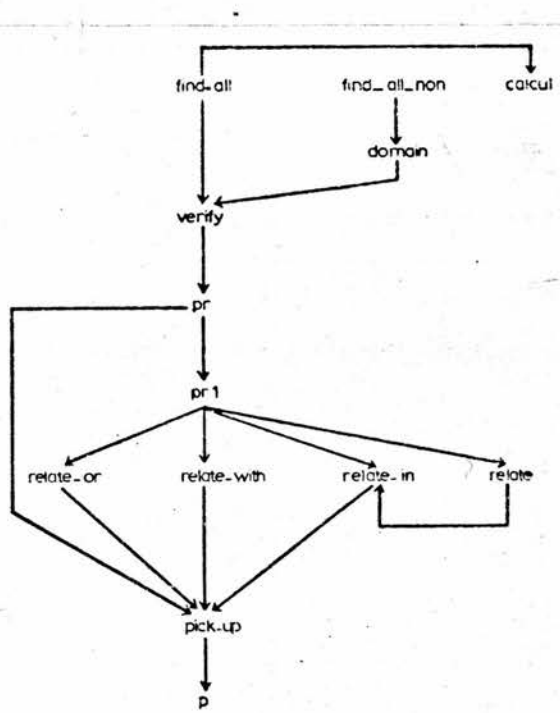


Fig.33

Verifying and finding procedures

-- 'which' questions with explicit quantification

Let us reconsider open questions for the case where the quantification is made explicit by the presence of an article.

Example: the sentence;

Quem e' um livreiro de "Artificial Intelligence"?
(Who is a publisher of "Artificial Intelligence"?)

has the following logical structure;

```
for([X,typ(pub(V))],
    and(pr(pub_of_tit(X,[artificial intelligence])),
        pr(set_equal(Y,X))),
    card(X,greater,0))
```

This question specifies a different type of search for information because the quantification structure 'for' is now present in the logical structure. The retrieval is accomplished by a verification procedure which calls the general procedure 'find_all',

```
find_all(
        [X,typ(pub(V))],
        and(pr(pub_of_tit(X,[artificial intelligence])),
            pr(set_equal(Y,X))),
        card(X,greater,0),
        B)
```

The computation of the logical structure now consists of verifying the term 'and'. Hence, the verification procedure is decomposed into two calls, one for each member of 'and'. When the individuals found

satisfy both routines of verification, a truth-value true is assigned to the overall verification. Otherwise, a truth value is assigned to the relating procedure, according to the logical matrix for 'and' (see Appendix 1):

and	t	f	u
t	t	f	u
f	f	f	u
u	u	u	u

t > f > u

t - true
f - false
u - undefined

t
↓
f
↓
u

The execution of 'find_all' can now be constrained because the cardinality of the expected individual is known: 'card(X,greater,0)'. The cardinality of the publishers to be found in the data base may be greater than zero, according to the definition of the indefinite article 'um' (a) which quantifies 'livreiro' (publisher). The cardinal corresponds to the third argument of procedure 'find_all', exemplified above.

This kind of constraint limits the search space by imposing two sorts of tests: the incompatibility and the compatibility tests. In the present example, the first one doesn't stop the search because the number of individuals to be found is different from one or zero. The second test assigns a truth value true or false, according to whether the number of individuals found is identical to the expected cardinality.

-- the treatment of conditional

We discuss other features of procedures 'find_all' and 'verify', not yet explained, by considering a statement.

Example: the sentence,

O tipo que escreve na "McGraw Hill" e' Nilsson.
(The man who writes for "McGraw Hill is Nilsson.)

has the following logical structure,

```
for([X,typ(V)],
    and(pr(author(X)),
        pr(pub_of_aut([mcgraw hill],X))),
    if(card(X,equal,1),
       pr(set_equal(X,[nilsson]))))
```

Statements, as opposed to open questions, require a single verification procedure. In the above example the presence of the definite article 'O' (The) imposes this particular quantification 'for'. The search for the individual 'X' is carried out by 'find_all',

```
find_all(
         [X,typ(V)],
         and(pr(author(X)),
             pr(pub_of_aut([mcgraw hill],X))),
         if(card(X,equal,1),
            pr(set_equal(X,[nilsson]))),
         B)
```

The second argument of 'find_all' is an 'and', and its verification follows the procedure explained in the previous example.

The third argument of 'find_all' is a conditional, and contains the definition of cardinality of article 'o'. The conditional is defined by the logical matrix for 'if' (see Appendix 1),

if	t	f	u
t	t	f	u
f	u	u	u
u	u	u	u

The conditional 'if' is implemented straighforwardly in the compatibility test through the procedure 'presuppose'. This procedure tests the compatibility of the list, representing the individuals found by 'find_all', with the expected cardinality present in the term 'if'. If the test is successful, the other property of 'if' is verified with truth value true or false. If not, the truth value undefined is assigned. For the above example, the cardinality of list 'X', corresponding to 'men who write for McGraw Hill', is 1 because there is only one member for 'X': Nilsson. Then, the compatibility is verified. As regards the incompatibility test, the conditional is inversed, ie. the resulting truth value is undefined if the incompatibility is verified.

-- the treatment of negation

The individuals verifying the negation of a logical structure are those in the complement set, with respect to the domain of individuals, of those verifying the structure. When the logical structure has no free variables, its negation consists only of inverting its truth value. This treatment of negation is accomplished in two different models of negation which depend on the sentence type, and determine two different data base searches. The first model is used for statements, closed questions and open questions with explicit quantification, and the other for open questions with implicit or presupposed quantification.

Let us consider the first model which follows "the psychological translation model" of negation (Clark,1970):

Example: the sentence,

Nilsson nao tem nenhum livreiro.
(Nilsson has no publisher.)

has the following logical structure:

```
not(for([X,typ(pub(V))],
        and(pr(pub_of_tit(X,_));
            pr(have(typ(V),[nilsson],typ(pub(V)),X)));
        card(X,greater,0)))
```

The negation operator 'not' covers the quantificational description. Therefore, we have a particular case for the verification procedure. The verification of a 'not' is performed by calling the verification of the quantificational structure 'for', and by inverting its truth value. The verification of a 'for' structure consists of executing the procedure 'find_all'. The inversion is done according to the logical matrix for 'not',

not	t	f	u
	f	t	u

Let us consider the second model which follows "the psychological true model" of negation (Trabasso,1970). For open questions with implicit quantification, the treatment of negation consists of getting all individuals that verify the structure with the truth value false. A formula is naturally considered false if something it asserts fails, in which case its natural denial is true.

Let us now consider a different example.

Example: the sentence,

> Quem nao escreve para "McGraw Hill" ou "North-Holland"?
> (Who does not write for "McGraw Hill" or "North-Holland"?)

has the following skeleton,

```
which([X,typ(V)],
not(
    pr(pub_of_aut(disj([mcgraw hill,north-holland]),X))))
```

where the quantifier 'for' is absent in the second argument, corresponding to the logical structure of the sentence. The predicate 'disj' stands for the disjunction. It determines, as the predicate 'coord' previously discussed, a certain kind of procedure for relating the arguments of the above 2-ary relation.

This sentence imposes a kind of search different from the previous example. The quantification is not explicit and its presupposition is solved by calling the procedure 'find_all'. However, as the description is now affected by the negation operator 'not', a new procedure 'find_all_non' is called to treat this kind of negation.

The procedure 'find_all_non' takes all individuals belonging to the domain of variable 'X', and verifies for each one whether the logical structure is true.

-- the treatment of meaningless

The treatment of meaningless sentences, which obtain the truth value undefined, is confined in TUGA to the presuppositions introduced only by definite articles.

Example: the following sentence

> Brown é o autor do artigo.
> (Brown is the author of the paper.)

makes a presupposition which is interpreted by TUGA with its current state of knowledge as undefined, because it

presupposes that there is one and only one paper (known to the system); and that this paper has just one author.

4.6.2. THE OUTPUT ASPECT

Answers are statements occurring in response to user requests. Each request type requires a specific answering mode. The normal form of these statements is simplified in TUGA, as is briefly shown in figure 34.

request form	answer form
open question	'and' of data items or a single data item, preceded, where appropriate, the preposition occurring in the request
closed question	standard message
statement	standard message
command	action followed by a dialogue

Fig.34

Request and answer forms

The answer form for open questions depends on their specific type, and is generated by procedure 'print';

print([K-D-G,X],B)

where 'K' is the syntactical case corresponding to the role played by the data items retrieved from the data

base and for the question.

Example: the procedure call of 'print' for question,

Que livro escreveu Winston?
(What book did Winston write?)

has the form;

print([dir-boo([])-_,[artificial intelligence]],t)

where 'dir' means that "artificial intelligence" plays the role direct complement in that question. Case 'dir' was introduced by verb 'escreveu' (wrote). When the verb in the question is a special one, 'K' has the value 'sub', corresponding to a subject; because the interrogative marker acts as a subject in the sentence.

Example: the procedure call of 'print' for the question,

Quem e' o autor de "Artificial Intelligence"?
(Who is the author of "Artificial Intelligence"?)

has the form,

print([sub-typ(aut([]))-_,[winston]],t)

where 'K' is 'sub'. For open questions with a 'howmany' syntactic marker, 'K' would be 'card' because this type of question requests the cardinality of some individuals.

The truth-value 'B' in the 'print' clause is 't' (true) for both types of open questions when they are supported by any verb except special verbs. In other cases, the truth value 'B' takes the value obtained by

the verification procedure.

The gender 'G' and the domain 'D' serve to make accurate the answer form, either by imposing an agreement between the proper noun(s) and the necessary article, or to insert a standard message.

The variable 'X' stands for the list of data items retrieved. The answer is standard whether 'X' is variable or the empty list. The answer is also pre-defined whether the value 'B' is false or undefined.

The answer form for closed questions and statements is pre-defined according to the truth value 'B', assigned by the verification procedure. The following figure illustrates the kinds of answer forms:

input form \ B	t	f	u
closed question (affirmative mode)	Sim. (Yes.)	Não. (No.)	Talvez haja confusão na sua frase. É ambigua, e portanto não consigo responder-lhe. (1)
closed question (negative mode)	É verdade. (It is true)	E falso. (It is false)	*
statement	Concordo. (I agree.)	Não concordo. (I disagree.)	A sua frase pressupõe outros factos, logo um contexto. Como não possuo informação sobre o que foi dito anteriormente, a sua frase é ambigua. E, portanto não consigo responder-lhe. (2)

(1) (There is, may be, confusion in your sentence. It is ambiguous and I can not answer you.)

(2) (Your sentence presupposes other facts, and a context. Your sentence is ambiguous because I have no information about what happened previously. And, I can not answer you.)

Fig.35

Kinds of answer forms

4.7. DIALOGUE ORGANIZATION

The objective of this section is to consider how the grammar of dialogues, presented in section 3.1, is implemented in TUGA. We shall refer to the program activities and how they are organized for running dialogues.

Asking for and interpreting user answers are two program activities performed during dialogues with users. Other activities, such as the interpretation of user requests and the computation of program answers were discussed in sections 4.5 and 4.6, respectively. Both activities are handled by the grammar of dialogues through the use of exchange patterns and dialogue models. The procedure 'ola' organizes the dialogue and starts numbering the contributions (see module CONVER, for more details).

An exchange pattern ('dialogue') is a pre-defined exchange between the program and the user. It is defined by a name and a number, and provided by a message and the number of the following contribution. It consists of a question of an expected form, followed by a simple dialogue. The question is constructed with the value of the message (eg. a proper noun). The simple dialogue is the standard way to interpret user contributions: the question-answer pair. As regards exchange patterns, the

user contribution expected is always an answer. Program questions are motivated by the content of user requests. For example, interrogating 'the name of a new category' and 'under what categories may it be placed' are generated when the user wants to create a new category in the classification system.

The exchange pattern is called by the grammar of dialogues through its name and number. In case of non acceptance of the program question by the user, the initiative for restarting the dialogue belongs to the user. But the new dialogue may be nested in the previous dialogue, as often occurs in the process of classifying new documents.

A dialogue model (eg. 'converse') is a suite of unconstrained exchanges between the program and the user. It generates detailed expectations about the next contribution, by having an ordering for calling exchange patterns which may be altered by user. The user may give several answers which need not be ordered. Also, he may modify his previous answers. The program uses the success or failure of its predictions to determine what role the user contribution plays in the dialogue. Whenever a dialogue model is activated, an appropriate exchange pattern is invoked, and the program poses a question to the user and interprets the user response. If a failure occurs, the program is able to come back to

the same topic. For example, during the classification of a document, the user may oppose the program and request information about the location of a category in the classification system. And, the program only accepts three titles of documents, known to its data base, and according to its classification method. Therefore, it goes on asking the user till it attains that limit, and skips any unknown title. But if the user gives up, the program restarts a new dialogue. These features are implemented either by using recursion or backtracking. Recursion is the ability of a procedure to contain a procedure call to another copy of itself. The declaration of the procedure, for the dialogue model in charge of the classification process, contains a procedure call which matches the name of another copy of the same procedure declaration. Counters control the process of recursion during the program asking for references in the classification process. And, the clause 'handle' deals with contradictions arising when a new document is archived. Numbers (1 and 0) are assigned according to existing contradiction or otherwise, and summed over the facts. If the result is non-zero, a failure arises forcing backtracking and the restarting of the process.

In the course of conversing, the program remembers the events, by storing conversational states containing the name of the participant, the number of the

contribution, and active information, such as questions or answers. The example below illustrates this.

Example: consider the dialogue described by the following history (the dialogue is presented in the next subsection 4.7.1 as an example of scenario C; numbers appended to the natural language contributions correspond to the conversational states);

T={<u,1,c11>,<p,2,a21>,<u,2,c32>,<p,3,q43>,
<u,3,a53>,<p,4,r64>,<p,4,q74>,<u,4,a84>,
<p,2,a92>}

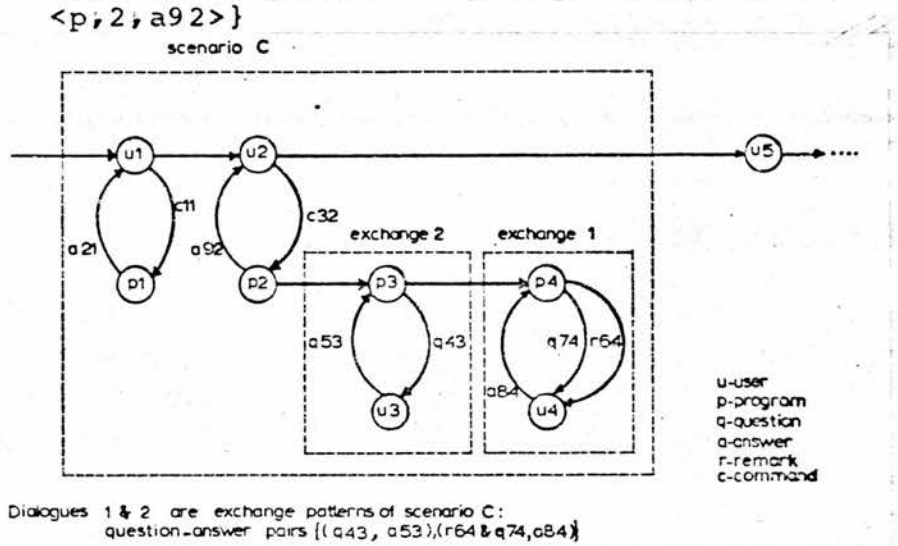


Fig.36

Conversational description

The program knows that the first user contribution, a command, <u,1,c11>, opens a dialogue composed of a simple question-answer pair (conversational state 1). The dialogue goes on with another user command, <u,2,c32>, which invokes a dialogue model for creating new categories. This dialogue model calls two exchange

patterns, and the dialogue is closed with the program answer <p,2,a92>. Note that this last event has the same conversational state number (2) as the event invoking the dialogue model. This information, shared by the program and one of its users, helps the program to decide on what to do and how to proceed. It chooses its course of action by inspecting previous user decisions, through the remembering mechanism.

The dialogue model, responsible for gathering information about new documents, exemplifies the use of backtracking. If the user changes his mind at any stage of the dialogue, the program backtracks to follow up the consequences of the new information. Supplied facts contradicting those already known are detected in the immediate interpretation of the user's input, and when this interpretation is complete, a failure leading to the restart of interpretation at a previous level occurs if a contradiction has been found. This mechanism overrides the repetition of unanswered questions (Mellish,1977), and skips questions by recognizing the content and form of the user's answers, as shown in the following dialogue.

.
.
.

p-Esta' bem!
Por favor, qual o nome do autor do documento?

(All right!
Please what is the name of the author of the document?)

u-Brazdil, DAI e 1978.

```
p-Por favor, qual e´ o tipo do      (Please, what is the
  documento?                          type of the document?)

u-Artigo.                                        (Paper.)

p-O novo documento no. 49,          (The new document no. 49,
  e a restante informacao                 and the remaining
  adicional foi arquivado: information has been stored:
  documento(
  paper(brazdil,49,experimental learning model,
        dai,1978,[111,1214],[1]))

  E que mais?                                (And what else?)
  .
  .
  .
```

This facility is achieved by special procedures dealing with facts and by grammar rules able to interpret the user´s answers.

The interpretation of the user´s short answers to program´s questions covers the following general situations:

i) user´s acceptance

ii) user´s refusal

iii) user´s change of mind

iv) user´s farewell

v) numerals

vi) single proper nouns (eg. author´s name)

vii) compound proper nouns (eg. document´s title)

where situations i), ii), iii) and iv) impose only reasoning, and situations v), vi) and vii) require a linguistic interpretation.

The following dialogues exemplify this capability by stressing four types of user answers (the token '=>' indicates these answers):

```
    .
    .
    .
  u-Arquive "Experimental          (Archive "Experimental
    learning model"!                   learning model"!)

  p-"experimental learning         ("experimental learning
    model" e' desconhecido!              model" is unknown!
    Gostaria de o arquivar!    I would like to archive it!
    Por favor, conhece a classificacao     Please do you
    de "Experimental learning      know the classification
    model"?                    for "Experimental learning
                                              model"?)

=>u-Sim, conheco.                          (Yes. I know.)

  p-Por favor, quais sao as        (Please what are the
    categorias do documento?           categories of the
                                             document?)

=>u-111 e 1214.
      .
      .
      .

  p-Por favor, de-me o titulo  (Please, give me the title
    de uma referencia do          of one reference from the
    documento em questao.           document in question.)

=>u-"Psychology of computer vision".

  p-"psychology of                 ("psychology of computer
    computer vision"                       vision" is known,
    e' conhecido, e esta' classificado          and it is
    nas categorias:          classified under categories:
    1214 -- "machine vision";
    1216 -- "information processing psychology", e
    1222 -- "modelling and representation of knowledge".
    Por favor, de-me o titulo de      Please, give me the
    outra referencia do          title of another reference
    documento em questao.              from the document in
                                       question.)

=>u-Nao tenho mais nenhuma.                (I have no more.)
      .
      .
      .
```

4.7.1. SCENARIOS

Scenarios are sets of expectations and presumptions regarding a certain type of situation. They are used in TUGA to classify the exchange patterns and organize their invocation. The embedded knowledge covers the ability

- to derive questions from relevant information or from the logical consequences of the information that is known about the questioned topic combined with general knowledge of the library world, and

- to handle the user's answers.

TUGA is a program able to play two roles in the library world. It acts as a librarian and as a library's secretary.

Possible events in the library world are grouped into the following scenarios:

Scenario A - information transaction

- Subscenario A1 - data output control
- Subscenario A2 - dictionary extensions

Scenario B - addition and/or deletion of data items

- Subscenario B1 - addition of new documents
- Subscenario B2 - addition of new categories
- Subscenario B3 - deletion of existing documents
- Subscenario B4 - deletion of classification categories.

Scenario C - classification category generation

Scenario D - document classification

Scenarios A, B and C may occur inside scenario D. Scenarios B2 and B4 may occur inside scenario C.

Scenario A, information transaction, covers the exchanges of information between the program and its users, and is supported by questions and statements.

Examples:

(1) u-Conhece alguma classificacao para "The CONNIVER reference manual"?
(Do you know any classification for "the CONNIVER reference manual"?)

p-Nao.
(No.)

(2) u-"Computational semantics" e "The process of question answering" sao dois livros.
("Computational semantics" and "The process of question answering" are two books.)

p-Concordo.
(I agree.)

(3) p-Por favor, qual o nome do autor do documento?
(Please what is the name of the author of the document?)

u-Brazdil.

Dialogues (1) and (2) are free exchanges, conducted under user initiative. Dialogue (3) is also attached to scenario D, document classification, and is conducted by the program. It consists of a question-answer pair, where the program question has a pre-defined form. The question form receives and handles information available in the previous user's answer.

Scenario B, addition and/or deletion of data items, deals with exchanges of information conveyed by user commands and program questions (the token '=>' indicates this type of questions):

Examples:

(1) u-Arquive "Experimental learning model"!
(Archive "Experimental learning model"!)

p-"Experimental learning model" e' desconhecido!
Gostaria do arquivar:
=>Por favor, conhece a classificacao do "Experimental learning model"?
("Experimental learning model" is unknown!
I would like to archive it!
Please, do you know the classification of "Experimental learning model"?)

(2) u-Apague "Experimental learning model"!
(Erase "Experimental learning model"!)

p-"Experimental learning model" e' conhecido e foi apagado!
("Experimental learning model" is known and has been erased!)

(3) p-O documento ficou classificado nas categorias:
312 -- "formal languages" e 1215 -- "natural language systems":
=>Por favor, deseja arquivar este documento na Base de Dados?
(The document has been classified under the categories:
.....
Please, do you want to archive it in the Data Base?)

u-Nao.
(No.)

These dialogues are conducted under user and program initiatives. Dialogues conducted by the program are particular, ie. program questions have pre-defined forms

which characterize the scenario.

Scenario C, classification category generation, covers exchanges conveyed by user commands followed by program answers or by particular dialogues.

Examples:

(1) u-Crie Prolog! (Create Prolog!)

p-Prolog e´ conhecido! (Prolog is known!)

(2) u-Quero criar uma categoria! (I want to generate a category!)

(3) p-Por favor, qual o nome =>da nova categoria? (Please, what is the name of the new category?)

u-"Pattern recognition".

p-"Pattern recognition" e´ desconhecida! ("Pattern recognition" is unknown!)

(4) =>Por favor, debaixo de que categoria a pretende inserir? Please, under what category do you want to insert it?)

u-"Machine vision".

p-"Machine vision" e´ conhecida! ("Machine vision" is known!)

(5) A nova categoria ficou inserida no sistema de classificacao e recebeu o numero 2141. The new category has been inserted in the classification system and received 2141 as a number.

=>E que mais? And what else?)

Scenario D, document classification, covers requests conveyed by user commands and followed by a set of particular exchanges, organized into dialogues.

Examples:

```
    .
    .
    .
u-Classifique "Explorations    (Classify "Explorations
  in cognition"!                        in cognition"!)

p-"exploration in cognition" e´      ("exploration in
  desconhecido!                 (cognition" is unknown!
=>Quer ainda classifica´-lo?      Do you still want to
                                        classify it?)

u-Esta´ bem.                              (All right.)

p-Vamos classifica´-lo!             (Let us classify it.
=>Por favor, de-me o titulo          Please, give me the
  de uma referencia do     title of one reference from
  documento em questao.     the document in question.)

u-Nao tenho nenhuma.                      (I have none.)

p-Esta´ bem!                                  (All right!
  Como nao consegui obter  As I have not received any
  nenhuma referencia, nao sei      reference, I don´t
  dar-lhe qualquer sugestao!         know how to give
                                   you any suggestion!
=>Quer ainda classifica´-lo?         Do you still want
                                       to classify it?)

u-Nao quero.                           (No, I don´t.)

p-Esta´ bem!                               (All right!
=>E que mais?                          And what else?)
    .
    .
    .
```

Exchange patterns are classified according to their use in these scenarios, as shown in the following figure. The classification is made possible through two of their arguments: name and number.

Scenario	Sub-scenario	Dialogue model	Exchange patterns
A	A1		asking whether the user wants more data
	A2	handling unaccepted sentences	asking the user's agreement for dictionary enlargement
			asking about a syntatic error in the sentence
			asking whether the unknown word is a proper noun
			asking which are the proper nouns
			asking about the unknown word gender
			asking about the unknown word domain
B	B1	addition of new documents	asking document author and publisher
			asking document date of publication and sort
			asking document categories
			asking the user to confirm his desire for document storage
	B2	classification system alteration	asking the category name under which the new one will be inserted
			asking the new category name
C			
D		classification method	asking the reference title
			asking the maximum of three categories
			asking the user to confirm his desire
			asking the user's choice for document categories
			asking the document classification

Fig.37

Classification of exchange patterns

CHAPTER 5

RELATED WORK

5.1. INTRODUCTION

TUGA is an "active question-answering system", that is it can run dialogues, based on natural language processing, by interchanging the iniative with the user. It also follows the logic programming approach. This entire perspective distinguishes TUGA from many other "passive question-answering systems" which support simple consultations, based upon isolated question-answer pairs, motivated by information retrieval.

The central feature of TUGA is the use of predicate logic as the sole programming tool for knowledge and data representation, deductive information retrieval and linguistic analysis. Compared with conventional programming languages, logic programs, written in Prolog, present a number of notable features, such as: pattern matching (unification); "multi-output" and "multi-input" procedures; "multi-purpose" procedures; "non-determinate" procedures; procedures may return

"incomplete" results; and, "program" and "data" are identical in form (Warren et al,1977).

The logic approach, based on work carried out mainly by (Kowalski,1974) and (Colmerauer,1973) since the early 1970's, is distinct from that adopted at Carnegie Mellon (Lesser et al,1974), Stanford (Walker et al,1975), and BBN (Woods,1972), where ambitious natural language understanding systems have been developed. The main difference between the logic programming approach and these other approaches is the following: logic supports a more uniform and higher level approach to natural language understanding than the other approaches, which require several programming languages and mechanisms to implement the different parts of a system.

5.2. OTHER PROGRAMS

A number of systems -- information retrieval systems, question answering systems and natural language understanding systems -- have been implemented over the last twenty-seven years. We will not attempt to review all of them, but it might be useful to compare TUGA with some of the more recent ones. We shall describe some of the techniques used and discuss their strengths and weaknesses. We have selected four systems which have influenced our design of TUGA. The first one followed

the logic programming approach. The last three were aimed, like TUGA, at practical implementation in relatively narrow knowledge domains. TUGA's knowledge domain, the library world, is for the first time considered in AI, as regards the construction of a program able to do more than mere information retrieval. A review of the AI history found only two related applications, although having mere information retrieval as the main feature (Levien,1965) and (Treusch,1975).

5.2.1. SDIBDE

SDIBDE {1} (Dahl,1977) is a natural language question answering system for accessing in Spanish a small data base of personnel information, following the Colmerauer approach.

SDIBDE served as the starting point for the evolution of TUGA: the actual text of Dahl's program was modified to deal with Portuguese, and with different worlds. Apart from the treatment of a different natural language, TUGA presents other significant differences from Dahl's program:

{1} This is our abbreviation for "Un systeme deductif d'interrogation de banques de donnees en Espagnol".

-- modular organization

Every module may be easily substituted when the problem world changes. During the development of TUGA three worlds were tested: personnel information, civil engineering legislation and the library service itself.

-- dialogue facilities

Interactions between the program and its users are governed by a grammar of dialogues, which covers also question answer exchanges.

-- other syntactical constructions

Examples: adverbial clauses, negative interrogative forms; sentence finals, coordinated statements and questions; coordination and disjunction of proper nouns; coordination of common nouns and adjectives, commands, and common users' answer constructions (eg. refusals, numerals).

-- indexing of grammar rules by some input words

The dictionary has been reformulated to exploit the automatic indexing provided by DEC-10 Prolog. It is a language independent module, defined as a set of unit clauses. The words covered by this feature are common nouns; proper nouns; verbs and adjectives. Indexing also saves time during the natural language analysis.

-- input checking of unknown words and compound nouns

The words of each input sentence are checked against the dictionary before calling the corresponding grammar rules.

-- incomprehensible sentence diagnosis

Sentences containing unknown syntactical structures or words are detected during parsing. The first ones are rejected as incomprehensible. The second ones are subjected to a dialogue in order to diagnose user's syntactical erro(s) or his ignorance about the presence of such word(s) in the program dictionary. For the last case the information on the unknown word may be asserted, and the sentence further analysed.

-- efficient production of logical structures

TUGA logical structures are easier to read than SDIBDE's, and closer to sentence interpretation. SDIBDE structures were longer and more complex because locations were reserved for syntactical functions. The truth-value 't' (true) was assigned to those argument positions when some optional constraint did not occur.

-- sophisticated data base facilities

TUGA data base is of relational type, and defined as a set of rules (non-unit clauses) and facts (unit clauses). This means that new facts may be easily added, giving

more generality to our approach.

TUGA evaluation of logical structures is different from SDIBDE's; and we consider it as more efficient and general. This difference depends on the organization of the data base itself and the definition of the role played by the notion of domain. TUGA data base is of relational type and SDIBDE's is not. Dahl considers a domain as a set of individuals, represented as a list; and we consider it as relation providing a way to access to a certain set of objects, one by one. For Dahl, domains are explicit, and for us implicit. On the other hand, Dahl uses the notion of domain in every evaluation, and we only use them for the treatment of negation. Apart from these advantages over the notion of domain, TUGA presents others concerning the handling of cardinality. The cardinality of a set is always computed in SDIBDE, but never in TUGA. This saves time when the data base is of medium or large size.

The treatment of Portuguese also necessitated the handling of the following linguistic features:

- improvement as regards the use of verb 'ser' (to be);
- use of verb 'ter' (to have);
- verb tense ellipsis (to be);
- subject agreement in the output;

- use of the relative/interrogative pronoun 'quem' (who) only with nouns that refer to humans;
- re-definition of some articles ('a'; 'no' and 'not all') and introduction of a new one ('the i');
- double negation imposed by the use of 'nenhum' (no);
- definite articles with proper nouns;
- disjunction of proper nouns; and
- conjunction of statements and questions.

5.2.2. LSNLIS

LSNLIS (Woods,1972;1976b) is a prototype of natural language question answering system for accessing a large data base of information about the moon rock samples collected during the Apollo 11 mission. It covers a rather wider subset of natural language than TUGA does.

LSNLIS natural language analysis is based on the general notion of computing a representation of the meaning of a phrase from representations of the meanings of its constituents. Firstly, input English questions are mapped into syntactic parse trees, by means of augmented transition networks (ATN's) (Woods,1970). Secondly, parse trees are translated into concrete specifications of what the computer is to do in order to answer the questions. The two types of information used in determining the meaning of a question, syntactical

information about sentence construction and semantic information about constituents, are therefore separated.

TUGA natural language analysis, based on Colmerauer's framework (Colmerauer,1977), considers that syntax and semantics are combined, and that no separate parsing phase is necessary. The grammar is expressed in logic by means of a formalism due to Colmerauer and Kowalski, which is a natural extension of context-free grammars. (Pereira&Warren,1978) argued that this formalism is clearer, more concise and (in practice) more powerful than ATN's, and at least as efficient as ATN's. They also showed that ATN's can be translated into this formalism.

The result of both analyses is different from a formal point of view. In Woods's, the meaning representation is a procedure specification, while it is a logic interpretation in Colmerauer's. However, there is a close parallel between the forms of the quantified expressions, Woods's FOR primitive and Colmerauer's three-branched quantifier, as it is stressed through their definitions.

In general, an instance of a quantified expression, according to Woods, takes the form:

(FOR <quant> X / <class> : (p X) ; (q X))

where <quant> is a specific quantifier, X is the variable

of quantification and occurs open in the expressions (p X) and (q X); <class> is a set over which quantification is to range; (p X) is a proposition that restricts the range, and (q X) is the expression being quantified.

Colmerauer's three-branched quantifier quant; introduced by an article α , creates a new formula quant(x,p,q); from a variable x and two formulae p and q; with the following reading: "for α x such that e1; it is true that e2", where e1 and e2 are the propositions corresponding to p (the one being quantified) and q (the one that restricts the range of x's).

When we attempt to compare the CPU times for parsing superficially similar sentences by both compiled systems; LSNLIS and a comparable TUGA's version, we get better results for TUGA. We list five examples with sentences of LSNLIS taken from (Burton,1976) and of TUGA. Note that the DEC KI-10 processor used for TUGA is merely nearly twice as fast as the KA-10 processor used for LSNLIS, and that timing data is averaged over 10 tests.

(1) Give me all analyses of S10046.
245 msec.

Quais sao os titulos da Elsevier?
(What are the titles of Elsevier?)
36 msec.

(2) How many breccias contain olivine?
175 msec.

Quantos autores escrevem para a Elsevier?
(How many authors write for Elsevier?)
38 msec.

(3) List modal plag analyses for lunar samples that contain olivine.
265 msec.

Quais sao os autores dos artigos que estao publicados desde 1977?
(Who are the authors of papers that have been published since 1977?)
56 msec.

(4) What is the average composition of olivine?
275 msec.

Qual e´ o escritor de "Artificial Intelligence"?
(Who is the author of "Artificial Intelligence"?)
40 msec.

(5) How many breccias do not contain Europium?
240 msec.

Quantos artigos nao sao titulos de Nilsson?
(How many papers are not titles of Nilsson?)
50 msec.

It is difficult to conclude upon these results, whether semantic interpretation while parsing (Colmerauer) is better than separating parsing from semantic interpretation (Woods), because Woods´s semantic analysis is more complex than Colmerauer´s one. In Colmerauer´s no mass nouns and comparatives are considered, and the use of semantics is directed only to select a logical structure.

The data base in TUGA is of relational type which has an important advantage over LSNLIS data base: the relational approach stresses data independence. This means that the user is isolated from the actual data base

organization, and has no need to define data accessing programs and the way data are internally organized.

5.2.3. PLANES

PLANES (Waltz,1975;1977;1978) is a natural language question answering system for accessing a large data base of information about naval aircraft maintenance and flight data.

PLANES natural language analysis is similar to LSNLIS except that the initial parse bypasses a syntactical parse tree representation in favour of a paraphrase expression of canonical phrases. This paraphrase is fed back to the user for confirmation before an interpretative phase maps the paraphrase into a concrete specification of what answer the program is to give the questions. LSNLIS is less tolerant than PLANES of nongrammatical requests because it is necessary to do a complete syntactical parse before performing semantic analysis. The PLANES data base is considerably larger and more complex, and the vocabulary and semantics of the PLANES world are consequently also somewhat larger than those of the LSNLIS world.

TUGA, like PLANES, is also a data independent system. The language dependent modules need not be substantially altered to accommodate an extended data

base or a new one (the major alteration is the dictionary). On the other hand, relations, properties and data structures need to be modified for a new knowledge domain. In PLANES, data is viewed as being divided into relations. Such a relational approach was also adopted in TUGA, with the advantage of using Prolog, a programming language typically used for the definition of relations (Emden,1978). This means that data definition and data manipulation languages are joined in a single programming language with a superior power and generality. There is no need for any particular query language and for a real relational data base. Logic programming allows the construction of virtual relational data bases, ie. explicit facts and rules are put together allowing the deduction of implicit information.

5.2.4. GUS

GUS (Bobrow et al,1976) is a dialogue program which plays the role of a travel agent in a goal-oriented conversation with a client. GUS has a modular architecture, and is composed of four interactive modules: morphological analyzer, syntactical analyzer, frame reasoner, and language generator. The frame module is built upon Minsky's notion (Minsky,1974) of a frame, a structure which can be instantiated to represent specific instances of events or entities.

GUS is only concerned with asking questions and understanding answers. It uses a system of travel-related frames to direct dialogue and instantiate memory representations of what it is told. The system asks questions to instantiate all of its frames with information provided by the client or inferred by the system. One type of inference made by GUS is generated by default assignments for certain frame fillers.

TUGA converses with users. It deals with asking questions and understanding answers, and understanding questions and producing answers (as any QAS). The knowledge-specific frames of the type used by GUS correspond to exchange patterns and dialogue models in TUGA. These procedures are controlled by an overall structure able to develop dialogues with users. Control structures displayed by TUGA are expressed naturally in Prolog without complicated programming concepts, which are, on the contrary, necessary for GUS (Mellish,1977). TUGA conversational power is obtained from a grammar of dialogues combined with Prolog's backtracking ability. The grammar articulates a system of frames and scenarios. The power is expressed by the possibility of several systems of dialogue, nested dialogues and the user's changes of mind in dialogues on gathering information.

CHAPTER 6

SUGGESTIONS FOR FURTHER RESEARCH

During the design and development of TUGA some experimentation was pursued on questions of efficiency and on possible solutions to present limitations. Some of these open points were explored and a solution was chosen, but have not been discussed until now. This section presents several points, grouped under general themes, such as representation of meaning, grammar extensions, evaluation and retrieval machinery and output generation, and details some already explored. By doing this kind of analytical search we approached the main aspects of further research, and we observed their interconnections. The following questions open the discussion:

1) What is the best representation of meaning?

2) What are the program improvements required to augment its conversational ability?

3) What are the main factors affecting the speed of response?

- Representation of meaning

There are currently two different approaches to representing the meaning of a sentence within the Colmerauer framework: 1) the use of a sole quantifier 'for', as one of the concepts for building up the definitions of articles, and 2) the use of complex quantifiers through the straightforward identification of articles. While (Colmerauer,1977), (Dahl,1977) and (Cotta&Silva,1978) adopted the first alternative, (Pasero,1976) and (Pique,1978) adopted the second one, but no reasons were put forward in favour of either one.

The two alternatives differ in the translation and evaluation of meaning of articles. By doing a previous translation of each article through 'for' we ensure a single machinery for its evaluation, while by not doing any discrimination we are obliged to have an evaluation mode for each article.

The first alternative, the use of 'for', allows an explicit interpretation of each article, but complicates the semantic representation of the sentence. On the other hand, it complicates the definition of the conditions of truth and its evaluation, namely as regards the undefined value. Also, it does not allow the straightforward identification of articles. The second alternative allows the straightforward identification of

articles and simplifies the evaluation of the truth conditions.

The kind of logical structure, obtained with the second alternative, the use of complex quantification, facilitates the construction of phrases as answers. The access to the meaningful components of the sentence and to articles is direct, and does not need to go through the intermediate step of using 'for' structures. The direct access makes this alternative more appropriate for anaphora resolution. Also, it suggests the terms 'pr', 'coord' and 'disj', appearing in our logical structures, are no longer needed (Pique,1978). However, Pique's relating operations and searching procedures do not seem more efficient than those available for the first alternative. In particular, the handling of lists in the arguments of a relation is not efficient. In spite of these considerations, it is not yet clear which of the alternatives is more efficient for medium or large data bases. Some experimentation is still required.

A way out of this controversy is the construction of a different logical system based upon a select choice of operators. This is what Colmerauer is doing now, retaking his experimentation already tested on lambda notation (Coelho,1977) and considering the epsilon notation (Coelho,1979a). This research strives for a clearer semantic representation, the implementation of

conjunction and disjunction and the suppression of free variables in logical structures. Such a representation will make it easier to discover the role played by each sentence fragment in building the whole meaning.

The limitations of the actual logical system, the inexistence of 'or' and 'entailment' definitions (Keenan,1972), render difficult the rigorous treatment of 'don't know' answers and the handling of incompletely specified information in the data base. However, these limitations are not easy to overcome on account of differences between logical and linguistic definitions. A solution would be the use of four-valued logic instead of three-valued logic. Each of the four states, corresponding to 'yes', 'no', 'undefined' and 'don't know', would be assigned to one truth value. However, a problem arises concerning that assignment, and on account of the ordering of truth values.

The implementation of 'don't know' answers was worked out upon a three-valued logic by subjecting the truth values to a condition. This condition is asserted in the data base during the evaluation of the logical structure, in particular when the relation giving access to unknown data items is evaluated. This condition is used for producing the appropriate answer. The unknown data items are represented by skolem functions.

Example: an answer to the question,

Quine esta´ publicado pela Elsevier?
(Is Quine published at Elsevier?)

is computed through the evaluation of the relation ´pub_of_aut(X,Y)´ against the document data base. Three hypotheses may occur as regards its evaluation: 1) Quine is published by Elsevier; 2) Quine is published by McGraw Hill; and 3) the publisher of Quine is unknown, but known to exist;

1) pub_of_aut(elsevier,quine)
2) pub_of_aut(´mcgraw hill´,quine)
3) pub_of_aut(pub_of_aut(quine),quine)

The unknown data item in 3) is represented by ´pub_of_aut(quine)´. And, to hypotheses 1), 2), 3) correspond the following answers 1a), 2a), 3a), respectively:

1a) Sim (Yes)
2a) Nao (No)
3a) Nao sei (I don´t know)

As the first and third hypotheses correspond to the evaluation of the relation with the truth value true, we distinguish them by asserting a conditional ´if(pub_of_aut,elsevier,quine)´ only for the third hypothesis, meaning ´if Quine is published by Elsevier then the truth value of the sentence depending on this

condition is true´.

- Grammar extensions

The grammar machinery deals with two types of forms: the discourse forms covering only aspects regarding the Portuguese language (Portuguese grammar), and the interaction forms covering aspects of the Portuguese language and other aspects concerning reasoning capabilities (grammar of dialogues).

The extension of the Portuguese grammar is required in order to cover new syntactical constructions, composite verbs, adverbs, plurality quantifiers, idiomatic expressions, pronominal reference and more elliptical constructions.

The grammar of dialogues will be improved by diversifying the type of sentences and by allowing knowledge acquisition during a dialogue.

As an example of new question types, we point to:

1) ´meta´ questions, about the dialogue itself and the knowledge processes,

2) ´why´ questions implying another feature, the content, for classifying questions, and a prover for deducing answers,

3) ´conditional´ questions for dealing with the

antecedent/consequent articulation and contributing for solving the reference problem,

4) 'indirect' questions and 'cleft' sentences for increasing user versatility in posing questions, such as

Que foi que Nilsson escreveu?
(What was it that Nilsson wrote?)

which is a version of the question,

Que escreveu Nilsson?
(What did Nilsson write?)

5) questions with elliptical constructions.

Usually, ellipsis is included in the general resolution of discourse anaphora (Nash-Webber,1978) because the machinery for dealing with it needs the same kind of syntactical and semantic data used for handling pronominal reference. One of the important points to consider is the kind of representation for sentence meaning. Another point is the kind of hypothetical reasoning needed for. We have been considering these two points by selecting two types of ellipsis: 1) the one appearing isolated in coordinative constructions, such as the ellipsis of argument, exemplified below,

Woods escreveu "Meaning and Machines" e esta' editado na "Academic Press".
(Woods wrote "Meaning and Machines" and it is published by "Academic Press".)

verb, complements, and verb plus complements; and 2) the one appearing immersed in a dialogue, such as,

u-Qual o autor de "Artificial Intelligence"?
(Who is the author of "Artificial Intelligence"?)

p-Nilsson.

u-E de "Computational Semantics"?
(And of "Computational Semantics"?)

The first question has an ellipsis of verb, e´ (is), which is already tackled by TUGA. The second question has an ellipsis of interrogative pronoun, verb and argument,

E (qual e´ o autor) de "Computational semantics"?
(And (who is the author) of "Computational semantics"?)

and it is branched to the previous question. We envisage solving the first type of ellipsis by using copy rules, supported on expected skeletons, and the second type by ad-hoc rules based on syntactical data.

As an example of new statement types, we point to 'conditional' statements.

The acquisition of new knowledge, facts and rules expressed in Portuguese during the dialogue involves new dialogue models. This acquisition may occur in pre-defined scenarios where the program is able to check whether the user is authorized to augment the data and knowledge bases. In fact, this acquisition implies situations where redundancies and contradictions between the new knowledge and the actual data bases may occur, and the program must protect its data against undue destruction. But, in order to take full advantage of natural language communication it is necessary to

implement a context machinery by allowing the user and the program to share fully the dialogue history, through remembering and association.

- Evaluation and retrieval machinery

The optimization of the evaluation of logical structure and the retrieval of data items is a factor for speeding answers. The use of parallelism and intelligent backtracking (Pereira&Monteiro,1978) is under study, whenever and/or operations are involved. The order of calling sub-expressions will be modified according to calling numbers of the terms involved (each variable is appended to the calling number of the term where the variable has been instantiated). It is also envisaged that the interaction of the data base relational scheme and context information will serve to guide the search. This implies the reformulation of the data base organization (eg. the use of the tree of domains for defining relations), and the existence of meta rules for guiding more appropriately the retrieval. The handling of presuppositions and the output of additional explanations also implies more work on the evaluation and retrieval machinery. Answers given by TUGA in meaningless and other particular situations are not sufficient for users. Some experimentation has been already done on answers for meaningless situations, on

simple answers to closed questions and statements, and on 'dont' know' answers. The experimentation suggested several modifications to retrieval procedures, as regards some grammar rules (eg. 'prin' rules) and as regards the output generation module. We found that some of these modifications needed further thought because they affect procedures belonging to different TUGA modules. In particular, some grammar rules do not at present allow, the transport of information required for that sort of explanatory answer. For example, closed questions and statements are translated without the transport of the information regarding their arguments. Moreover, the information regarding gender and number is not used in retrieval operations.

- Output generation

Closed questions, requiring straightforward answers of yes/no kind, and statements need to be supported with information, available in the logical structure after its evaluation. A simple grammar able to synthetize that information in a phrase is under development, and it already operates for structures with one article. The data extraction is more difficult when several articles are present in the sentence, on account of the complexity of the logical structure.

A final note about program efficiency concerns the analysis of incorrect sentences and lexical search. As a matter of fact, a previous diagnosis of the words involved in the input sentence avoids an impossible analysis, and speeds up the answering for this kind of sentence. On the other hand, lexical recognition previous to the selection of the appropriate grammar rule for articles, verbs, nouns and adjectives, avoids unnecessary backtracking and access to inappropriate rules. The access to the good rule is speeded up by indexing. These improvements on program efficiency are already implemented in a new version of TUGA.

CHAPTER 7

CONCLUSION

The objective of designing the program TUGA was the development of a feasible method for consulting and creating data bases in natural Portuguese.

This is the first time Portuguese has been used in such a practical application. Indeed, within Computational Linguistics, the only work on Portuguese seems to be (Machado-Holsti,1976), which was aimed at building up a model of a generative transformational grammar. Our work extends the interest of man-machine communication in natural language, by showing that it need not be restricted to English, the language most widely studied and adopted.

The resulting program allows dialogues where the program and its users behave in the way humans normally do in a dialogue setting. Users can ask questions, provide answers and issue commands in a natural and convenient way, without bothering excessively with the form of the dialogues and sentences.

The natural language communication is achieved by an approach which supports the organization of dialogues at a higher level than Colmerauer's framework. The approach aims at the description of the history of dialogues and the structuring of single participant contributions into models of dialogue. It allows the writing of grammars of dialogue, combining general purpose exchanges with special ones closer to the application chosen.

Colmerauer's framework governs the communication below sentence level by supporting the translation of isolated sentences into logical structures. It was chosen because it provides a clear representation of meaning and because it is amenable to a computational approach. Moreover, it is easily adapted to Portuguese and to different applications.

Logical structures are evaluated according to an algorithm that differs from those of Colmerauer and Dahl. The algorithm is based on Colmerauer's hypotheses, but it is modified to fit relational data bases and to improve searching efficiency.

This approach has made possible the development of an applied AI program for solving certain problems in the library domain, in a manner which is acceptable to users in the domain. The program includes a knowledge acquisition facility which enables users to augment or modify the knowledge base, those allowing them to improve

the program's behaviour in future consultations, and to produce better document classifications. An important aspect of TUGA is that it does not demand great expertise of the user. This application, if developed, could be useful for the AI community because it involves a natural mode of communication based upon (two-way) dialogue.

The approach that has been taken stresses the use of predicate logic to represent knowledge. An example is the implementation of TUGA in Prolog. We chose this programming language on account of its outstanding features, such as closeness to natural language, modularity, one language for program and data, logical variable, computation of relations, economy (the Prolog interpreter occupies only about 12k on the DEC-10, while the Conniver system needs about 51k), understandability, learnability and inbuilt search strategy. These features enable us easily to adapt the program to other knowledge domains, because modifying clauses in Prolog is easier than modifying procedures in other programming languages.

To conclude, we would like to mention three wider implications of this study. First, logic is relevant from a practical point of view. This has been brought out by the use of TUGA as a computer aid in support of the teaching of logic in secondary schools. The program gave the pupils an explicit understanding of the use of logical machinery as a tool for conversing in Portuguese

with a computer. The aspects covering representation of meaning and inferential capabilities served as concrete examples, instead of typical sentences far from reality. Second, handling the understanding of natural language is a matter of importance. It requires from the computer the burden of comprehending, and supporting greater access to computers for naive users. This means that it is possible to avoid any specialized mediator between the computer and users, and in particular for those applications requiring immediate responses. Third, our work is a basis for other applications in Portuguese. An example is the consultation of the civil engineering legislation which is now available at the LNEC library computer terminal. This application was carried out by two university students who adopted our Portuguese grammar machinery and built a knowledge base and a dictionary for that domain.

ACKNOWLEDGEMENTS

I would like to thank the stimulating interaction of my friend Luis Pereira for initiating me in the field of Artificial Intelligence and challenging me constantly through hours and hours of technical and philosophical discussion.

My utmost recognition goes also to David Warren, my close supervisor and thesis advisor, who taught me to conduct this research, encouraged me to proceed along these last three years and gave fruitful advices about the preparation of this manuscript. I am grateful for his many extensive and critical comments which facilitated the final rewriting of this thesis.

I am indebted to Fernando Pereira, Pavel Brazdil and Chris Mellish for reading and commenting on an earlier version of this thesis.

To Jose Cotta and Antonio Silva, for many helpful discussions during the writing of this thesis, and particularly on Chapter 6.

To Carlos Morais, head of Informatics Department at the National Laboratory for Civil Engineering (LNEC), in Lisbon, where great part of this work was carried out, goes my deepest appreciation for having given me the opportunities and support which made it possible.

To Joao Cunha and Luis Cunha, my colleagues at LNEC, for carrying out some of my duties and thereby giving me with more time to pursue my research.

Thanks are also due to LNEC, and in particular to its director, Ing. Ferry Borges, for incentive and support during the course of this work.

Finally, I want to express my gratitude towards INVOTAN, from whom I received a grant during my periods at the University of Edinburgh and the funds which permitted my matriculation.

APPENDIX 1: THE LOGICAL SYSTEM FOR REPRESENTING PORTUGUESE

We follow closely Colmerauer's system (Colmerauer, 1977). Changes are made for the vocabulary of the syntax and for the definition of articles.

I) - Summary of the Syntax

(1) Vocabulary

terms:

variables:	x	with x ∈ X
constants:	k	with k ∈ K
compound terms:	coord(x)	with x ∈ X
	disj(l)	with l as a list of x ∈ X
statement formulae	: ei, si, ni	
relational symbols	: r	with r ∈ R
quantifier	: those(x,e)	
logical connectives	: and, not	

Individual constants are chosen to treat proper nouns. Compound terms are chosen to treat coordinated proper nouns and the disjunction of proper nouns. Relational symbols are chosen to treat verbs, adjectives

and nouns.

Note that the meaning of those(x,e) is: "those x's which satisfy statement e", and that it is only considered combinations formed by two rules: conjunction and negation.

(2) Syntactic rules

The grammatical sentences of this formal language are:

(a) a statement formula ei has one of the 6 forms:

1) r(si1,si2,...,sin) with r ε R and degree(r) = n

2) and(e1,e2)

3) if(e1,e2)

4) not(e)

5) equal(n1,n2)

6) greater(n1,n2)

(b) a set formula si has one of the 3 forms:

1) c with c ε K

2) x with x ε X

3) those(x,e) with x ε X

(c) an integer formula ni has one of the 2 forms:

1) j where j is an integer such that j >/0

2) card(si)

definition 1 - the occurrence of an individual variable x in a formula f is free if it does not arise inside a sub-formula of the form those(x,e).

definition 2 - a formula which contains no free individual variable occurrences is closed.

II) - Summary of the Semantics

Having formalized the language, we formalize the notions of situation and truth, the necessary apparatus for determining the truth-value of any closed formula.

definition 3 - a situation g is an application which, to each relational symbol r ε R of degree n, associates a n-ary relation p=g(r), of which the arguments ki are individuals or the subsets of the set of proper nouns K, and of which the value p(k1,k2,...,kn) is either "true", "false" or "undefined", according to the values of the ki.

definition 4: let g be a situation:

the value val(ei) of a closed statement formula ei is defined by :

(1) if ei = r(si1,...,sin) then
val(ei) = p(val(si1),...,val(sin))
with p = g(r)

(2) if ei = and(e1,e2) then
val(ei) = min(val(e1),val(e2))
with true > false > undefined

(3) if ei = if(e1,e2) then
if val(e1) = true then val(ei) = val(e2)
if val(e1) =/= true then
val(ei) = undefined

(4) if ei = not(e) then
if val(e) = true then val(ei) = false
if val(e) = false then val(ei) = true
if val(e) = undefined then
val(ei) = undefined

(5) if ei = equal(n1,n2) then
if val(n1) = val(n2) then val(ei) = true
if val(n1) =/= val(n2) then
val(ei) = false

(6) if ei = greater(n1,n2) then

if val(n1) > val(n2) then val(ei) = true

if val(n1) < val(n2) then

val(ei) = false

the value val(si) of a closed set formula si is defined by

(1) if si = c with c ∈ K then

val(si) = (c)

(2) if si = x where x has been substitued by the formal representation of a set E then

val(si) = E

(3) if si = those(x,e) then

val(si) = {the union of all the subsets E of K:

$$val(e_{x<--E}) = true$$

where $e_{x<--E}$

represents the formula e in which a formal representation of the set E has been substituted for every free occurrence of x }

the value val(ni) of a closed integer formula ni is defined by :

(1) if ni = j where j is a non negative integer then

val(ni) = j

(2) if ni = card(sj) then

val(ni) = number of elements of the set val(sj)

The semantics of a closed statement, set or integer formula f are simply the variations of its values val(f) in different situations.

N.B. This logical system presents a unique quantifier,

those(x,e)

which allows to explicit the domain of variable x. This feature does not belong to first-order predicate logic where variable x is not restricted, and its implicit domain remains always the same. Also, this feature allows to operate upon predicate logic and to establish a better definition for articles.

III) - Definition of articles

The procedure for operating upon the well-formed formulae of this logical system demand they be set in a convenient form, by appealing to a sole quantifier. Thereby, a new quantifier 'for' is introduced, and defined in terms of quantifier 'those(x,e)',

for(x,e1,e2) = e2
x <-- those(x,e1)

ie. the value of 'for' is equal to the value of e2 when we substitute all free occurrences of x in e2 by

those(x,e1).

The quantifier 'for' allows the explicitation of the domain of a variable. This feature improves the definition of articles, and in so doing contributes to a better translation of natural language sentences. The quantifier 'for' records the difference of meaning of the articles, and expresses them in terms of only some well defined concepts.

The following equalities establish the definitions of the articles considered in our subset of Portuguese.

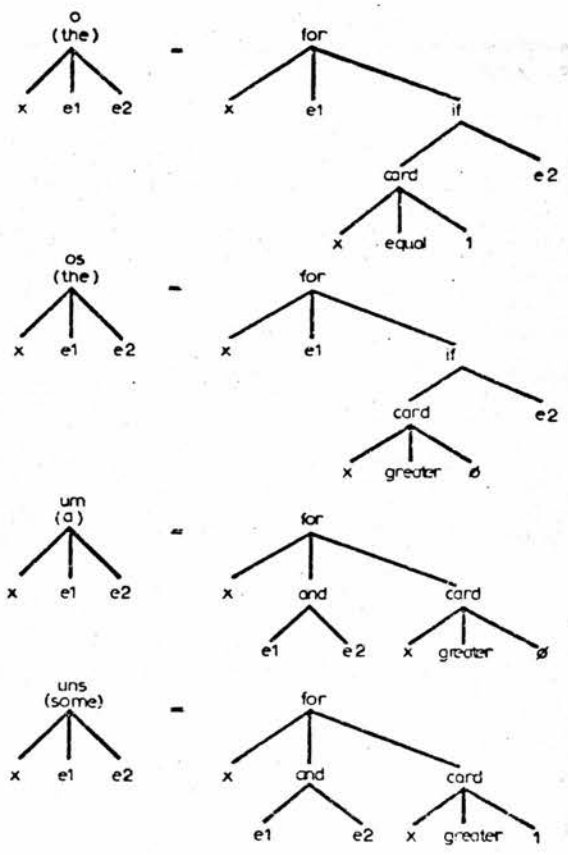


Fig.38

Equalities for articles

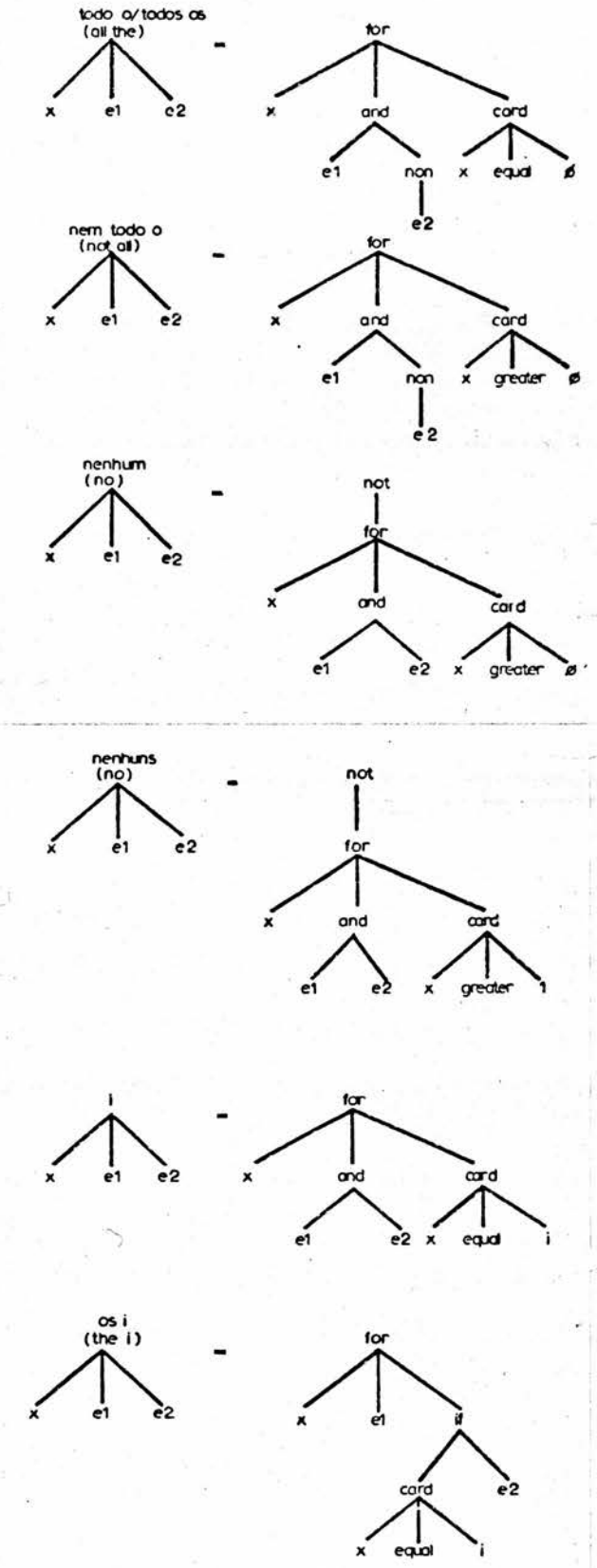


Fig. 39

Equalities for articles

APPENDIX 2: A BRIEF SURVEY OF PROLOG

Prolog is a simple but powerful programming language founded on symbolic logic, developed at the University of Marseille (Roussel,1975), as a practical tool for logic programming (Kowalski,1974;Colmerauer,1975;Emden,1975). A major attraction of the language, from a user's point of view, is ease of programming. Clear, readable, concise programs can be written quickly with minimum error (Coelho et al,1979). Recently, an efficient compiler and an interpreter were implemented on the DECsystem-10 (Warren,1977a). A user's guide is already available (Pereira et al,1978).

Like Lisp, Prolog is an interactive language designed primarily for symbolic data processing. Both are founded on formal mathematical systems -- Lisp on the lambda calculus and is typically used for the definition of functions, Prolog on a powerful subset of classical logic (Tarnlund,1977) and is typically used for the definition of relations. Pure Lisp in fact can be viewed as a specialization of Prolog (Warren,1977b).

1. SYNTAX

Here is a Prolog program, consisting of two clauses, for specifying the concatenation relation of two lists:

```
concatenate([],L,L).
concatenate([X¦L1],L2,[X¦L3]):-concatenate(L1,L2,L3).
```

In general, a Prolog program consists of a set of procedures, where each procedure comprises a number of clauses. The procedure name is called a predicate ("concatenate" above), and has an arity which is the number of its arguments (3 above). A clause begins with a head or procedure entry point, and continues with a body. If the body is not empty it is separated from the head by ":-" (2nd clause above). Every clause terminates with a ".". The head diplays a possible form of the arguments to the procedure's predicate. The body consists of a number (possibly zero) of goals or procedure calls, which impose conditions for the head to be true. If the body is empty we speak of a unit clause (1st clause above).

In general, all Prolog objects are terms. A clause is a term, a predicate or a goal with their respective arguments, and the arguments themselves are terms. For example, a tree of 5 categories of a classification system is represented by the following term,

```
t(t(void,linguistics,void),
  'computing sciences',
  t(t(void,ai,void),applications,t(void,software,void)))
```

A term is either a variable (distinguished by an initial capital letter), an atom ("void" above), or a compound term. A compound term comprises a functor ("concatenate" or "t" above) of some arity N > = 1, and a sequence of N terms as its arguments ("t(void,ai,void)" above). An atom is treated as a functor of arity 0. A term of the form [H¦T] stands for the list .(H,T), whose head is H and tail is T. The empty list is denoted [], and a list with exactly two elements by [A,B].

The second clause above is just infix notation for the term

```
:-(concatenate([X¦L1],L2,[X¦L3]),
   concatenate(L1,L2,L3))
```

where ":-" is a binary functor. The functor ":-" takes as arguments the head and the body of the clause. If a body has more than one goal the comma separating the goals is just another binary functor used in infix notation. The above term stands for a clause because it figures in the set of clauses for a procedure. It is dintinguished by a final ".".

Apart from syntax conventions, the names and arities of terms (and their number) are arbitrary, except for a pre-defined set of procedures which are built into the implementation of the language, and which achieve input, output, arithmetic, etc.

2. SEMANTICS

Prolog differs from most programming languages in that there are two quite distinct ways to understand its semantics. The procedural or operational semantics is the more conventional, and describes as usual the sequence of states passed through when executing a program. In addition a Prolog program can be understood as a set of descriptive statements (one for each clause) about a problem. The declarative or denotational semantics, which Prolog inherits from logic, provides a formal basis for such an understanding. Informally, one interprets terms as shorthand for natural language phrases by applying a uniform translation of each functor, eg.

```
void = "the empty tree"

t(L,N,R) = "the binary tree with root N, left
            subtree L and right subtree R"
```

A clause "P :- Q,R,S." "where P,Q,R and S are metavariables standing for terms, is interpreted as

"P if Q and R and S"

A clause "P." is interpreted as "P is true".

Each variable in a clause should be interpreted as some arbitrary object (ie. variables are universally quantified). The type of the object conveyed by a

variable will be appropriate to the functor(s) where the variable figures by using terms in a consistent way throughout the program.

The declarative semantics simply defines (recursively) the set of terms which are asserted to be true according to a program. A term is true if it is the head of some clause instance and each of the goals (if any) of that clause instance is true; where an instance of a clause (or term) is obtained by substituting, for each of zero or more of its variables, some term for all occurences of the variable.

Thus the only instance of the goal: -

```
concatenate([a],[b],L).
```

is

```
concatenate([a],[b],[a,b]).
```

It is the declarative aspect of Prolog which is responsible for promoting clear, rapid, accurate programming. It allows a program to be broken down into small, independently meaningful units (clauses), and it allows some understanding of a program without looking into the details of how it is executed.

3. PROCEDURAL SEMANTICS

It is the procedural semantics that describes the way a goal is executed. The objective of execution is to produce true instances of the goal. It then becomes important to know that the ordering of clauses in a program, and of goals within a clause, which are irrelevant as far as the declarative semantics is concerned, constitute crucial control information for the procedural semantics.

To execute a goal, the system searches for the first clause whose head matches or unifies with the goal. The unification process (Robinson,1965) finds the most general common instance of the two terms, which is unique if it exists. If a match is found, the matching clause instance is then activated by executing in turn, from left to right, each of the goals of its body (if any). If at any time the system fails to find a match for a goal it backtracks, ie. it rejects the most recently activated clause, undoing any substitutions made by the match with the head of the clause. Next it reconsiders the original goal which activated the rejected clause, and tries to find a subsequent clause which also matches the goal. Execution terminates if no goals remain to be executed (the system has then found a true instance of the original goal). Backtracking may then be invoked to find other true instances of the goal. Execution fails

when no true instances of the original goal are found, and terminates if it cannot find any more true instances. Termination however cannot be guaranteed, even if there are no more true instances (eg. if there are infinite branches).

Note that the execution just defined is a left to right depth-first process. Note also that because unification always provides the most general common instance between a goal and a matching clause, all the most general true instances of a goal can potentially be found (ie. apart from termination issues).

Basically, each execution step is justified by Robinson's Resolution Principle (Robinson,1965). This principle subsumes in a single inference rule the classical rules of "modus ponens" and "instantiation" in formulations of first order predicate calculus. For example, from;

```
p(X):-q(a,X),r(X).
```

and

```
q(Y,f(Y,Z)):-s(a,Z).
```

it allows to conclude

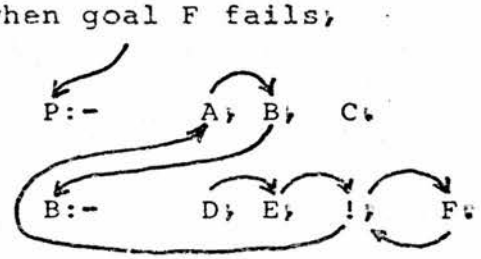
```
p(f(a,Z)):-s(a,Z),r(f(a,Z)).
```

by "execution" of q(a,X).

Besides the ordering of clauses and the sequencing of goals within clauses Prolog provides just one other essential mechanism for specifying control information. This is the "cut" symbol, written "!". It is inserted in a program just like a goal, but it is not to be regarded as part of the logic of the program and should be ignored as far as the declarative semantics is concerned.

The effect of the "cut" is as follows: when first encountered, as a goal, "cut" succeeds immediately. If backtracking should later return to the "cut", the effect is to fail the goal which caused the clause containing the "cut" to be activated. In other words, the "cut" operation commits the system to all choices made since execution of the goal activating the clause begun, ie. other alternatives for that goal are not considered, as well as for all goals occurring in the matching clause before the "cut". By means of a "cut" one can ensure that some goals, once partly executed by a clause up to a "cut", either must continue that partial execution or fail. The "cut" renders deterministic the whole partial execution made by the activated clause up to it.

Example of the effect of a "cut" in the flow of control, when goal F fails,



where A, B, C, D, E, F, and P are metavariables standing for predicate instances.

If F fails, backtracking returns to goal A, immediately before B, the goal that activated the clause with the "cut".

4. OUTSTANDING FEATURES OF PROLOG

Let us briefly review the combination of features which make Prolog a powerful but simple to use programming language.

(1) A declarative semantics inherited from logic in addition to the usual procedural semantics.

(2) Identity of form of program and data - clauses can be employed for expressing data, and can be manipulated as terms by interpreters written in Prolog.

(3) The input and output arguments of a procedure do not have to be distinguished in advance, but may vary from one call to another. Procedures can be multi-purpose.

(4) Procedures may have multiple outputs as well as multiple inputs.

(5) Procedures may generate, through backtracking, a sequence of alternative results. This amounts to a high level form of iteration.

(6) Terms provide general record structures with any number of fields. An unlimited number of record types may be used, and there are no type restrictions on the fields of a record.

(7) Pattern matching replaces the use of selector and constructor functions for operating on structured data.

(8) Incomplete data structures may be returned (ie. structure containing free variables) which may later be filled in by other procedures.

(9) Prolog dispenses with go to, do for and while loops, assignment, and references (pointers).

(10) The procedural semantics of a syntactically correct program is totally defined. It is impossible for an error condition to arise or for an undefined operation to be performed. This totally defined semantics ensures that programming errors do not result in bizarre program behaviour or incomprehensible error messages.

(11) No part of the program is concerned with the details of the underlying machine or implementation.

APPENDIX 3: LISTING OF TUGA

conver.

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*          A PROGRAM CONVERSING IN PORTUGUESE                   */
/*              PROVIDING A LIBRARY SERVICE                      */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/



/* Operators used in program writing */

:-op(500,xfy,'.').
:-op(300,xfy,'-').
:-op(500,fx,'?').


/*+++++++++++++++++PROGRAM ORGANIZATION+++++++++++++++++*/


/*  conver : control of dialogues
    input  : input recognition
    output : answer and question program generation
    gramma : fragment of Portuguese grammar
             syntax, semantics and core vocabulary
             (articles, pronouns, prepositions,
             adverbs and conjunctions)
    dictio : peripheral vocabulary
             (nouns, verbs and adjectives)
    dbms   : data base managment system
    dbase  : data and knowledge bases                        */


/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/


/*  Opening and general control of dialogues
    between the program and its users                        */

ola:-opening,
     ( ?conv(N),M is N+1,
      opening1,converse(M) ;
      opening2,converse(1) ).
```

```
/* Conversational modes:                    */

/*   model  of general dialogues            */

converse(N):-dialogue(N,S),
        ( ?fr(u,N1,fecho), +conv(N), -fr(u,N1,fecho),! ;
         continue(N,M), converse(M) ).


continue(N,M):-what_else,
               ( ?cv(P), M is P+1 ; M is N+1).


/*   models of particular dialogues */


converse(K-K1,N-M,S,C,D,CON):-repeat,( ?dia(T);T=N),
               dialogue(K,K1,T,S),
               (( ?fr(u,T,nouns(C));
                  ?fr(p,T,title(A,B)),(var(A),C=B;C=A)),
                M is T+1, -all(dia(_)),
                (K=arq,handle(K-D,S,C,CON);true),!;
                 P is T+1, +dia(P),fail).



converse(K,N,LR,C,P):-var(N),?dial(M),N=M,
                       converse(K,N,LR,C,0).
converse(K,N,[],[],3):-!.
converse(K-K1,N,LR,C,P):-dialogue(K,K1,N,S),!,
                         course(K-K1,N,LR,C,P).


course(K,N,[L1,..LN],[C,..L],P):-check_if(K,N,T,C),
        (var(C),C=[],S is N+1,converse(K,S,LN,L,P);
         M1 is N+1,P1 is P+1,(?rf(O),O1 is O+1,+rf(O1);
         +rf(P1)),
         doc(L1,T),!,converse(K,M1,LN,L,P1)).

course(K,N,[],[],P):- ?fr(u,N,recusa),
                       converse(K,N,[],[],_),!.

course(K-K1,N,LR,C,P):-( ?fr(u,N,desvio),!,fail;
           shift,N1 is N+1,dialogue(K,3,N1,R),
           ( ?fr(u,N1,recusa),!,fail;
            return,N2 is N1+1,converse(K-K1,N2,LR,C,P) )).


converse(N,Z,d(A,D,L,Y,C,R)):-converse(arq-4,N-N1,
                    categorias,C,_,_),
                    +dial(N1),converse(clas-1,M,R,_,_),
                    N2 is M+1,converse(N2,d(D,A,L,Y,Z)).
```

```
converse(N,d(D,A,L,Y,Z)):-repeat,
          ask(arq-1,N-M,autor,[A],D),
          ask(arq-1,M-M1,livreiro,[L],D),
          ask(arq-2,M1-M2,'a data de publicacao',[Y],D),
          ask(arq-2,M2-_,'o tipo',[Z],D), +cv(M2),
          -all(fact(_,_,_)).


converse(N,n(R,C)):-N1 is N+1,
             converse(lexi-2,tr(N1-N2,C,aceita)),
             dialogue(lexi,3,N2,C),N3 is N2+1,
             ?nn(L),converse(lexi-6,tr(N3-N4,L,aceita)),
             turn(N4,L).


converse(K-K1,tr(N-M,S,R)):-repeat,
                    ( ?cd(T);T=N),dialogue(K,K1,T,S),
                    ( ?fr(u,T,R),M is T+1, -all(cd(_)),!;
                      P is T+1, +cd(P),fail).


turn(N,[]):- +ok,N1 is N-1, +cv(N1).
turn(N,[L1,..L]):-converse(lexi-4,N-N1,L1,[G1],_,_),
           gen(G1,G),converse(lexi-5,N1-N2,L1,[P],_,_),
           dom(P,D),archive_pn(L1,G,D),turn(N2,L).


conversel(D,M):-(?cv(N);N=1),N1 is N+1,
                (D=ref([]),S=documentos,!;S=categorias),
                repeat,
                dialogue(said,1,N1,S),decide(D,N1,M).


/*  test in the document classification scenario */

check_if(clas-_,N,T,C):- ?fr(p,N,title(T,C)).


/* decisions to be taken in the information
   transactions scenario                          */


decide(_,N,s):- ?fr(u,N,aceita),!.
decide(_,N,n):- ?fr(u,N,recusa),!.
decide(_,N,M):- ?fr(u,N,number(M)),!.
decide(D,N,_):- +cv(N),conversel(D,N).


/* Subsidiary mechanisms of dialogues */


ask(_,M-M,Tipo,[V],N):- ?fact(N,Tipo,V),V==f(_),!.
```

```
ask(K-K1,M-M1,Tipo,V,N):-repeat,
                         question(K-K1,M-M1,Tipo,V,N,CON),
                         !,CON=0.


question(K-K1,N-M,Tipo,V,D,CON):-
                converse(K-K1,N-M,Tipo,_,D,CON),!,
                dont_insist(D,Tipo,V,CON),!.

dont_insist(N,T,[V],0):-!, ?fact(N,T,V).
dont_insist(_,_,_,_).

handle(arq-N,T,[],0).
handle(arq-N,T,[L1,..LN],R):-care_of(N,T,L1,S),
                              handle(arq-N,T,LN,Q), R is S+Q.


care_of(N,T,V,0):- ?fact(N,T,V),!.
care_of(N,T,V,0):- ?fact(N,T,f(D)),
                   -fact(N,T,f(D)), +fact(N,T,V).
care_of(N,_,V,C):-(integer(V),T='a data de publicacao';
                   (nol(V,liv,_);
                    nol(V,art,_),T='o tipo');
                   pn(V,_,D),
                   (D=typ([]),T=autor;
                    D=tit([]),T=titulo;
                    D=typ(pub([])),T=livreiro)),
                   ( ?fact(N,T,V1), -fact(N,T,V1),
                     +fact(N,T,V),C=1; +fact(N,T,V),C=0).
care_of(N,T,f(_),0):-(T=autor,D=typ(V);
                      T=livreiro,D=typ(pub(V));
                      T='a data de publicacao',D=yea([])),
                      +fact(N,T,f(D)).
care_of(_,_,_,1):-write('Houve um engano da sua parte!'),
                  nl.



/* Simple question-answer dialogues:
   between the user and the program */

dialogue(N,S):-user(S),nl,program(N,S).


/*  Exchange patterns between the program and its user */


/* in the document classification scenario */

dialogue(clas,1,N,S):-please,write('de-me o titulo de '),
       ( ?rf(K),K>=1,write('outra');write('uma')),
       write(' referencia do documento em questao.'),
       nl,dialogue(N,S).
```

```
dialogue(clas,2,N,S):-
       please,
       write('faca a sua escolha tendo em atencao'),
       write(' as categorias sugeridas'),
       nl,write('e as que julgar mais apropriadas.'),nl,
       write('De-me 3 categorias no maximo!'),nl,
       dialogue(N,S).

dialogue(clas,3,N,R):-
             write('Quer ainda classifica''-lo?'),
             nl,dialogue(N,R).

dialogue(clas,4,N,R):-
              write('De-me 3 categorias no maximo!'),nl,
              dialogue(N,S).

dialogue(clas,5,N,T):-please,
                   write(' conhece a classificacao de '),
                   write('"'),write(T),write('" ?'),nl,
                   dialogue(N,S).


/* in the information transactions scenario */

dialogue(said,1,N,S):-write('Quer mais '),
                      write(S),write('?'),nl,
                      dialogue(N,R).


/* in the classification category generation scenario */

dialogue(crie,1,N,R):- please,
  write(' debaixo de que categoria a pretende inserir?'),
  nl,dialogue(N,S).

dialogue(crie,2,N,R):-
           please,
           write(' qual o nome da nova categoria?'),nl,
           dialogue(N,S).


/* in the storage and erasure of data items scenario */

dialogue(arq,1,N,S):-
            please,
            write(' qual o nome do '),write(S),
            write(' do documento?'),nl,
            dialogue(N,_).

dialogue(arq,2,N,S):-
            please,
            write(' qual e'' '),write(S),
            write(' do documento?'),nl,
            dialogue(N,_).
```

```
dialogue(arq,3,N,S):-
            please,
            write(' deseja arquivar este documento '),
            write('na Base de Dados?'),nl,
            dialogue(N,_).

dialogue(arq,4,N,S):-please,
                     write(' quais sao as '),write(S),
                     write(' do documento?'),nl,
                     dialogue(N,R).


/* in the dictionary enlargement subscenario */

dialogue(lexi,1,N,_):-
        write('Existe algum erro sintatico na escrita'),
        write(' da sua frase?'),nl,
        dialogue(N,_).

dialogue(lexi,2,N,R):-( R=1,
 write('A palavra desconhecida e'' um nome proprio?');
 write('Alguma das palavras desconhecidas e'' um nome'),
 write(' proprio?')),nl,
 dialogue(N,_).

dialogue(lexi,3,N,R):-( R=1,
 write('Qual a palavra que e'' nome proprio?');
 write('Quais as palavras que sao nomes proprios?')),nl,
 user(S),sent(S,O), +nn(O).

dialogue(lexi,4,N,R):-write('Qual e'' o genero de '),
                      write(R),write(' ?'),nl,
                      dialogue(N,_).

dialogue(lexi,5,N,R):-write('A qual dos tipos, '),
     write('autor, titulo, livreiro, ou categoria,'),nl,
     write(' pertence a palavra '),write(R),write(' ?'),
     nl,dialogue(N,_).

dialogue(lexi,6,N,R):-
        write('Vou perguntar-lhe informacoes sobre '),
        write(R),nl,
        write('de forma a inseri-la no dicionario.'),nl,
        write('Concorda?'),nl,
        dialogue(N,_).


/* Analysis of user sentences:
      case of comprehensible sentences */

program(N,S):-prin(F,S,[]),+fr(u,N,F),
              answer(N,F), +fr(p,N,F),!,nl.
```

```
/*    case of incomprehensible sentences */

program(N,S):-sent(S,O),diagnostic(R,O,[]),
              omission,omission(N,R),
              ( ?ok, ?cv(N1),N2 is N1+1,
              write('A resposta `a sua pergunta sobre '),
              output4(R),write(' e'':'),nl,
              program(N2,S) ; +fr(p,N,incompreensivel)).



:-end.




input.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                    INPUT RECOGNITION                         */
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/



/* Input recognition character by character, and
   identification of words that compose user sentences */


user(F):-get(C),words(C,F),!.



/* Construction of simple and compound words */

words(C,[P,..PS]):-(characterl(C,34),wordl(C,C1,[C,..L]);
                    character(C,A),word(A,C1,L) ),
                   name(P,L),words(C1,PS).

words(44,[',',..PS]):-get(C1),words(C1,PS). /*comma */
words(63,[?]):- +interrogative.        /*question mark*/
words(46,[.                            /*full stop*/
words(33,[!]).                         /*exclamation mark*/
words(_,P):-get(C),words(C,P).

word(C,C1,[C,..CS]):-get0(C2),
                      (character(C2,A),word(A,C1,CS);
                       C1=C2,CS=[]).
```

```
word1(C,C1,[C,..CS]):-get0(C2),
                      (character1(C2,34),get(C3),C1=C3,CS=[];
                       character1(C2,C3),word1(C3,C1,CS)).


/* List of characters */

character(32,_):-!,fail.        /* space */
character(63,_):-!,fail.        /* ? */
character(46,_):-!,fail.        /* . */
character(33,_):-!,fail.        /* ! */
character(44,_):-!,fail.        /* , */
character(45,_):-!,fail.        /* - */
character(40,_):-!,fail.        /* ( */
character(41,_):-!,fail.        /* ) */
character(58,_):-!,fail.        /* : */
character(59,_):-!,fail.        /* ; */
character(10,_):-!,fail.        /* line-feed */
character(13,_):-!,fail.        /*    CR     */

/*   upper-case test  */

character(C,A):-(C=<90, C>=64, A is C+32 ; A=C).

/*  upper-case test for compound nouns  */

character1(C,A):-(C>=64,C=<90, A is C+32 ; A=C).


/* Diagnostic of the user sentence */

diagnostic(R) --> [],({ ?lex(R), -all(lex(_)),! ; R=[] }).
diagnostic(R) --> lexicon, diagnostic(R).



/* removal of separators */

sent([],[]).
sent([S1,..S],O):-separator(S1),sent(S,O).
sent([S1,..S],[S1,..O]):-sent(S,O).



separator(!).
separator(',').
separator('-').
separator(.).
separator(?).
separator(e).
separator(ou).
```

```
/* lexicon search */


lexicon --> [N], {nol(N,_,_)}.
lexicon --> [N], {pn(N,_,_)}.
lexicon --> [V], {vel(V,_,_);vel(V,_,_,_)}.
lexicon --> [A], {adl(A,_,_)}.
lexicon --> articles.
lexicon --> pronouns.
lexicon --> adverbs.
lexicon --> [P], {prep(P)} ; c,[C], {prep(C)}.
lexicon --> conj_c.
lexicon --> nb(N).
lexicon --> expressions.
lexicon --> [N],({ ?lex(R),+lex([N,..R])},!;{+lex([N])}).



:-end.




ouput.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*       ANSWER AND QUESTION PROGRAM GENERATION                 */
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/



/* Generation of responses according the user
   sentence type                                                */


answer(N,aceita).
answer(N,recusa):-agreement.
answer(N,desvio):-change.
answer(N,fecho):-close.

answer(N,number(M)).
answer(N,nouns(L)).
answer(N,title(T,C)):-
            (var(T),answer(N,order(pr(category([C])))),!;
            answer(N,order(pr(category([C],[T]))))).
```

```
answer(N,fact(conj(O1,O2))):-
                write('Para a primeira proposicao: '),
                answer(N,fact(O1)),!,
                write('E, para a segunda proposicao: '),
                answer(N,fact(O2)).

answer(N,fact(O)):-verify(O,B),response2(B).

answer(N,order(pr(O))):-O=..[categoria,C],var(C),
                        N1 is N+1,!,
                        dialogue(crie,2,N1,R).

answer(N,order(O)):-verify(O,B),!,answer1(N,O,K,T,B),
                    (order(N,K,T,B);true).

answer(N,question(conj(O1,O2))):-
                write('Para a primeira pergunta:'),
                answer(N,question(O1)),!,
                write('E, para a segunda pergunta: '),
                answer(N,question(O2)).

answer(N,question(O)):-verify(O,B),
                ( -negative,response3(B);response1(B)).

answer(N,which((G-sin)-D-X,K,O)):-(verify_d(X,O,_),!;
                        verify(O,B) ),print([K-D-G,X],t).
answer(N,which((G-_)-D-X,K,O)):-
                find_all([X,D],O,card(_,_,_),_),
                print([K-D-G,X],t).
answer(N,which((G-_)-D-X,O)):-verify(O,B),
                              print([sub-D-G,X],B).

answer(N,how_many([G-X,D],O)):-
                find_all([X,D],O,card(_,_,_),_),
                print([card-D-G,X],t).
answer(N,how_many(G-D-X,O)):-verify(O,B),
                             print([card-D-G,X],B).

answer(_,_):-nl,note.


/* Auxilliaries for the response generation */

answer1(N,pr(P),clas,T,B):- P=..[categoria,C,T],
                            response4(N,clas,C,T,B).
answer1(N,pr(P),info,T,B):- P=..[info,I,T],
                            response4(N,info,I,T,B).
answer1(N,pr(P),crie,C,B):- P=..[categoria,C],
                            response4(N,crie,C,_,B).
answer1(N,pr(P),dest,_,B):-P=..[destroi,C],
                           response4(N,dest,C,_,B).
answer1(N,pr(P),arq,T,B):-P=..[arquive,T],
                          response4(N,arq,T,_,B).
```

```
/* Types of user commands */

order(N,clas,[T],f):-classify(N,R,C,M),(var(C),!;
                     P is M+1,archive(P,T,C,R)).

order(N,crie,[C],f):-generate_cat(N,C,M).

order(N,dest,_,f).

order(N,arq,[T],f):-archive(N,T).


/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                  PRINT OF RESPONSES                         */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/


/*   General responses */

opening:-write('Ola''.'),nl,nl.
opening1:-return,request.
opening2:-start,request.


what_else:-write('E que mais?'),nl,nl.
start:-write('Vamos iniciar a conversa!'),nl.
return:-write('Vamos retomar a conversa anterior!'),nl.
change:-write('Ok.Vamos desviar-nos desta conversa!'),nl.
shift:-write('Desvia''-mo-nos da conversa!'),nl.
close:-write('Ok,esta conversa terminou!'),nl,
       write('Adeus, e ate'' `a vista!'),nl.


omission:-write('Nao compreendo esta frase ').
omission(N,[]):-reason,suggestion, +cv(N),!.
omission(N,R):-reason(R,C),
               (remark(N), ?fr(u,N,aceita),suggestion,!;
                suggestion(N,R,C)).


reason:-write('porque a sua construcao sintactica'),
        write(' e'' desconhecida!'),nl.

reason(R,C):-write('porque '),card(R,C),
             (C=1,write('a palavra '),output4(R),
              write(' e'' desconhecida!');
              write('as palavras '),output4(R),
              write(' sao desconhecidas!')),nl.


suggestion:-write('Corrija a sintaxe ou construa'),
            write(' outra parafrase.'),nl,
            write('Repita a pergunta por favor!'),nl.
```

```
suggestion1:-write('Desculpe, mas tem de reformular'),
             write(' a sua frase'),
             write(' empregando outras palavras!'),nl.

suggestion(N,R,C):-converse(N,n(R,C));suggestion1.


remark(N):-please,
      write(' preste atencao `a frase que escreveu!'),nl,
      dialogue(lexi,1,N,_).


request:- please,
       write(' escreva factos, ordens ou perguntas.'),nl.

please:-write('Por favor,').

agreement:-write('Esta'' bem!'),nl.


note1:-write('Talvez haja confusao na sua frase.'),nl,
       write('E'' ambigua, e portanto'),
       write(' nao consigo responder-lhe.'),nl.


note:-write('Compreendi esta frase.'),nl,
      write('No entanto,nao consigo encontrar'),
      write(' uma forma de responder-lhe.'),nl.



/* Responses for:
    closed user questions */


response1(t):-write('Sim.'),nl.
response1(f):-write('Nao.'),nl.
response1(u):-nl,note1.



/*  user statements */

response2(t):-write('Concordo.'),nl.
response2(f):-write('Nao concordo.'),nl.
response2(u):-
    write('A sua frase pressupoe outros factos,'),
    write('logo um contexto.'),nl,
    write('Como nao possuo informacao sobre'),
    write(' o que foi dito'),nl,
    write('anteriormente, a sua frase e'' ambigua.'),nl,
    write('E, portanto, nao consigo responder-lhe!'),nl.
```

```
/*    interrogative negative user sentences */

response3(t):-write('E'' verdade.'),nl.
response3(f):-write('E'' falso.'),nl.
response3(u):-nl,notel.


/*  user commands occurring in the pre-fixed scenarios */

response4(N,info,[L],[T],t):-
            write('"'),write(T),write('"'),
            write(' e'' conhecido,'),nl,
            write('e possui a seguinte informacao:'),nl,
            outputl(doc(V),L),nl.

response4(N,clas,[C],[T],t):-
          write('"'),write(T),write('"'),
          write(' e'' conhecido,e'),
          write(' esta'' classificado nas categorias:'),
          nl,output(cat([]),C),nl.

response4(N,crie,[C],_,t):-write('"'),write(C),write('"'),
                           write(' e'' conhecida!'),nl.

response4(N,crie,[C],_,f):-
         (var(C),write('Proponha outro nome!'),nl,!,fail;
          write('"'),write(C),write('"'),
          write(' e'' desconhecida!'),nl).

response4(N,dest,[C],_,t):-write('"'),write(C),write('"'),
              write(' e'' conhecido, e foi apagado!'),nl.
response4(N,dest,[C],_,f):-write('"'),write(C),write('"'),
            write(' nao pode ser destruido, porque e''),
            write(' desconhecido!'),nl.

response4(N,arq,[T],_,t):-write('"'),write(T),write('"'),
         write(' e'' conhecido e nao ha'' necessidade '),
         write('de o arquivar!'),nl.
response4(N,arq,[T],_,f):-write('"'),write(T),write('"'),
                    write(' e'' desconhecido!'),nl,
                    write('Gostaria de o arquivar!'),nl.

response4(N,_,_,[T],f):-write('"'),write(T),write('"'),
                      write(' e'' desconhecido!'),nl.


/*    open user questions:
      proper nouns as response */


print([K-_-G,[]],B):-ifyes(B),nought(K-G),nl.
print([K-_-G,[X]],f):-var(X),nought(K-G),nl.
```

```
print([_,X],u):-write('A sua pergunta e'' indefinida!'),
 nl,(var(X),
 write('Nao encontro nenhuma entidade que a satisfaca.'),
 nl;true).
print([card-D-_,X],t):- (X=coord(Y);X=Y),card(Y,C),
                         output(D,[C]),nl.
print([K-D-G,[X]],t):-nonvar(D),D=doc(V),outputl(D,X).
print([K-D-G,X],t):- ( X=[Y-_], output2(Y) ; true),
           (X=[_-L],impk(K,D,[L]),cond_output(D,[L]);
            (X=coord(L);(D=cat([]);D=ref([])),X=[L];X=L),
             impk(K,D,L),cond_output(D,L)),nl.


impk(prep(em),yea([]),_):-output2(em).
impk(prep(em),_,L):-(L=[N];L=[N,..._]),pn(N,G,_),
                    (G=mas,output2(no);output2(na)).
impk(prep(por),_,L):-(L=[N];L=[N,..._]),pn(N,G,_),
                     (G=mas,output2(pelo);output2(pela)).
impk(prep(P),_,_):-output2(P).
impk(_,_,_).


/* Data output  */


cond_output(D,L):-card(L,C),
        (C<6,output(D,L),!;
         get_till_5(L1,L,L2,_),card(L2,C2),
         (D=ref([]),write('Os cinco primeiros');
          write('As cinco primeiras')),write(' sao:'),nl,
          output(D,L1),nl,
          write('Existem mais '),write(C2),
          (D=ref([]),write(' documentos!');
           write(' categorias!')),nl,
          conversel(D,M),output3(D,L2,C2,M) ).

output(ref([]),[]):-write('Nao existem referencias!'),nl.

output(D,[A]):-outputl(D,A),write('.'),!.
output(D,[A,L]):-outputl(D,A),write(' e '),outputl(D,L),
                 write('.').
output(D,[A,..L]):-outputl(D,A),write(', '),output(D,L).


outputl(D,K-N-X-Y-Z-W-V):-nonvar(D),D=doc(V),
          write(K),write(' no.'),write(N),write(','),nl,
          write('autor: '),write(X),write(','),nl,
          write('livreiro: '),write(Y),write(','),nl,
          write('ano de publicacao: '),write(Z),
          write(','),nl,write('classificacao: '),
          output(cat([]),W),(V=[],write('.'),!;
           write(','),nl,
           write('referencias: '),output(ref([]),V),
           write('.'),nl ).
```

```
outputl(D,X):-
 (nonvar(D),(D=cat([]),(cat(C,X,_,_);cat(_,_,C,X));
 D=ref([]),(book(_,X,C,_,_,_,_),nl,write('livro no. ');
 paper(_,X,C,_,_,_,_),nl,write('artigo no. '))),
 write(X),write('--'),write('"'),write(C),write('"') ;
 nonvar(D),call(not(integer(X))),
 (D=tit(V);D=typ(pub(V))),write('"'),write(X),write('"');
 write(X)).


output2(X):-write(X),write(' ').


output3(_,_,_,n):-!.
output3(D,L,_,s):-(D=ref([]),write('Os');write('As')),
                  write(' restantes sao: '),output(D,L),!.
output3(D,L,C,N):-N>=C,output3(D,L,_,s),!.
output3(D,L,_,N):-get_till_n(L1,L,_,N),output3(D,L1,_,s).


output4([A]):-outputl(_,A),!.
output4([A,B]):-outputl(_,A),write(' e '),outputl(_,B).
output4([A,..L]):-outputl(_,A),write(', '),output4(L).


/* Handling of nothing as response */

nought(card-mas):-!,write('Nenhum.'),nl.
nought(card-fem):-!,write('Nenhuma.'),nl.
nought(prep(em)-_):-
        write('Que eu saiba, nao foram publicados'),
        write(' pela mesma entidade.'),nl.
nought(prep(P)-_):-write(P),write(' ninguem.'),nl.
nought(_):-
   write('Nenhuma entidade satisfaz a sua pergunta.'),nl.



:-end.




gramma.


/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*            FRAGMENT OF PORTUGUESE GRAMMAR                   */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
```

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                  SYNTAX AND SEMANTICS                         */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/


/*    Grammar re-writing rules;its function    */


/* Main sentence types:    */

/*   user responses and remarks  */

prin(fecho) --> [adeus], final(_,_).

prin(O) --> (nega(_,non(_)), {O=recusa};
             afir(_,yes(_)), {O=aceita} ),
            (ve(afir,_), atrib ; {true} ), final(_,_).

prin(desvio) --> ve(des,_), ( [P], {prep(P)} ; {true} ),
                 ( no(conv,_) ; {true} ), final(_,_).

prin(number(M)) --> ve(afir,_), atrib,
            art(X,_,_,for(_,_,card(_,equal,M))),
            noun([X,..._],_), final(_,_).

prin(title(T,C)) --> [N],( {not(integer(N)),
                  pn(N,_,tit(V)),T=N;
                  pn(N,_,cat([])),(N<1500;N>10000),C=N}),
                  final(_,_).

prin(nouns(L)) --> (nouns(L);
                nega(_,non(_)),ve(sei,sin),{L=[f(d)]} ),
                final(_,_).


/*    user questions, statements and commands */

prin(P) --> pre_locl, prop(O), final(P,O),!.

prin(X2) -->  ( { ?interrogative } ;  pre_loc2 )  ,
              int_rell(interrog(O,X2),X1), prop(O),
              final(_,interrog),!.

prin(P) --> prop(O), final(P,O),!.


/* Sentence final */

final(_,interrog) --> [?], { -interrogative } ; [!],
                      { -diga }.
final(question(O),O) --> [?],{ -question ; true }.
final(question(O),O) --> [.],{ -question }.
```

```
final(fact(O),O) --> [.].
final(order(O),O) --> [!].


/* Phrase(s) */

prop(P) --> nucleus(L,O1,O2,O), compls(L,O1,O2),
            ({P=O} ;
             comma,[e], prop(P1), {P=conj(O,P1)} ).


nucleus(L,O1,O2,O) --> [nucleus(L,O1,O2,O)],!.
nucleus(L,O1,O2,O) --> arg(sub-Y,O4,O), neg(O3,O4),
                       (verb([Y,..L],O1), {O2=O3};
                       {L=[noun-A-D-Z]}, ve(ter,_),
                       {O1=set_equal(X,Z)},
                       advg_g(A1-D-Z,prep(de)-Y,L,O2,O3) ).
nucleus(L,O,O1,O) --> verb(L,O) , (per_pron ; {true}).


/* Negation */

neg(O,O),[verb(X,O1),P,none(G-N)] -->
                    [nao], verb(X,O1), [P], none(G-N),!.
neg(O,O),[verb(X,O1),none(G-N)] --> [nao],
                    verb(X,O1), none(G-N).
neg(O,non(O)) --> [nao], {negative}.
neg(O1,O) --> [neg(O1,O)].
neg(O,O) --> [].


negative:- ?negative,!.
negative:- +negative.


nega(O,non(O)) --> [nao].


/* Affirmation */

afir(O,yes(O)) --> [sim];[ok];[por,favor];[sim,por,favor].
afir(O,O) --> [].


/* Subject placement at the beginning of the sentence */

int_rel1(I,X) --> int_rel2(I,X).
int_rel1(I,X1),[nucleus(L,O1,O4,O),mov_arg(X2,O2,O3)] -->
                    int_rel2(I,X1),
                    arg(X2,O2,O3), nucleus(L,O1,O4,O).
```

```
/* Subject, interrogative article and relative
            pronoun generation                          */

int_rel2(I,X),[arg(K-X,O,O)] --> pron(I,K-X).
int_rel2(interrog(O,X2),X1),
          [case(K),art(X1,O1,O2,and(O1,O2))] --> case(K),
                                    interrog_art(K-X1,O,X2).


int_rel2(rel,A-X),
              [case(K),s_nucleus([A2-X2,X1],O3,O2,O1,O),
               mov_arg(prep(de)-A-X,O,O)] --> case(K),
               pron(rel,prep(de)-A2-X),
               def_mark(A2),
               s_nucleus([A2-X2,X1],O3,O2,O1,O).


/* Complements */

compls([X,..L],O1,O) --> [mov_arg(X,O2,O)],
                         compls(L,O1,O2).
compls([],O,O) --> [].
compls([K-A-D-X,..L],O1,O) --> compls(L,O1,O2),
                               arg(K-A-D-Y,O2,O),
                               {arg_form(K,Y,X)}.

compls(L,O1,O) --> [compls(L,O1,O)].


/* Arguments(subjects) 1 */

arg(X,O1,O) --> [arg(Y,O1,O)],!, {X=Y}.


/* Subject inversion and coordenative conjunctions */

arg(sub-A-D-X,O1,O2),[neg(O3,O1),ve(ter,N)] -->
                      neg(O3,O1), ve(ter,N),
                      inv_mark, arg(sub-A-D-X,O1,O2).

arg(sub-A-D-X1,O1,O2),[neg(O3,O1),verb([A-D-X1,X2],O4)]-->
                       neg(O3,O1),
                       verb([A-D-coord(X1),X2],O4),
                       inv_mark, arg(sub-A-D-X1,O1,O2).


/* Arguments(subjects) 2 */

arg(K-X,O1,O) --> case(K), n_phrase(X,O1,O).
```

```
arg_form(prep(de),Y,coord(Y)):-var(Y),!.
arg_form(_,Y,Y).


/* Noun phrase */

n_phrase((G-N)-D-X,O,O) --> (proper_nouns((G-N)-D-X);
                  def_art(G-sin), proper_nouns((G-N)-D-X)).
n_phrase(X,O1,O) --> s_nucleus([X,..L],O3,O2,O1,O4),
           compls(L,O4,O), com, relatives(X,O2,O3),[end].


s_nucleus(L,O3,O2,O1,O) --> [s_nucleus(L,O3,O2,O1,O)].

s_nucleus([X,..L],O3,O2,O1,O) --> art(X,O3,O1,O),
                          com, adjs(X,O4,O5),[end],
                          common_noun([X,..L],O5,O6), com,
                          adj_g(X,O6,O2),[end].


/* Relative phrases of restrictive type */

relatives(X,O2,and(O2,O3)) --> [com(C)], int_rell(rel,X),
                     prop(O4), end(C), relatives(X,O4,O3).

relatives(X,O2,O2) --> [].



/* Disjunction and conjunction of proper nouns */

proper_nouns((G3-N)-D-disj([I,..L])) --> [I],{pn(I,G1,D)},
                           end(disj),{accord(G1,G2,G3)},
                           pns((G2-plu)-D-L), [end].

proper_nouns((G3-N)-D-[I,..L]) --> [I], {pn(I,G1,D)},
      end(C), {accord(G1,G2,G3)}, pns((G2-N)-D-L), [end].


pns((G3-plu)-D-[I,..L]) --> [com(C)],
               (def_art(_-sin);{true}),
               [I], {pn(I,G1,D)},
               end(C),{accord(G1,G2,G3)}, pns((G2-N)-D-L).
pns((fem-sin)-D-[]) --> [].


accord(mas,_,mas).                accord(fem,G,G).


/* Common nouns */

common_noun(X,v,O) --> noun(X,O).
```

```
common_noun(X,O1,and(O1,O2)) --> noun(X,O2).


/* Conjuntion of nouns */

nouns([L1,..LN]) --> [L1], {nom(L1)}, end(_),
                     nos(LN), [end].

nos([L1,..LN]) --> [com(C)], [L1], {nom(L1)},
                   end(C), nos(LN).
nos([]) --> [].

nom(X):-pn(X,_,_).
nom(X):-nol(X,_,_).



/* Adjectives  and adjectival groups */

adjs(X,v,v) --> [].
adjs(X,O,O) --> [com(C)], adj([X],O), end(C).
adjs(X,O1,and(O1,O3)) --> [com(C)], adj([X],O1),
                          end(C), adjs(X,O2,O3).


adj_g(X,O,and(O,O3)) --> [com(C)], adj([X,..L],O1),
                  compls(L,O1,O2), end(C), adj_g(X,O1,O3).

adj_g(X,O,O) --> [].


atrib --> (adv_m ; adv_q ; {true} ), none(_).
atrib --> adv_m;adv_q.
atrib --> [].



/* Adverbial groups */

advg_g(X,Y,_,O1,O) --> adv_g(X,Y,O1,O).
advg_g(X,Y,L,O1,O), [compls(L,O2,O3)] --> compls(L,O2,O3),
                                          adv_g(X,Y,O1,O).

adv_g(X,Y,O1,O) --> adver, s_nucleus([X,Y,..L],v,_,O1,O2),
                    compls(L,O2,O).



/* Ellipsis and generation of an article
                  according the case          */

inv_mark,[case(sub),none(G-N)] --> none(G-N).
inv_mark --> [].
```

```
def_mark(A),[def_art(A)] --> [].


case(K) --> [case(K)].

case(sub),[none(G-N)] --> none(G-N).
case(sub),[not_all(A)] --> not_all(A).
case(sub) --> [].

case(dir),[ind_art(A)] --> [].
case(dir) --> [].

case(noun),[art(X,O,v,O)] --> [].
case(noun),[art(X,O,v,O)] --> ind_art(A),!.
case(noun) --> [].

case(prep(P)) --> contract,[P],{prep(P)}.


/* Interrogative and relative pronouns general rules */

pron(interrog(O,which(X,O)),sub-X) --> int_prons.
pron(interrog(O,which(A-typ(V)-X,O)),sub-A-typ(V)-X) -->
                    int_pronsl.
pron(interrog(O,which(X,K,O)),K-X) --> int_pron(K).
pron(interrog(O,which(A-typ(V)-X,K,O)),K-A-typ(V)-X) -->
                    r_i_pron(K).
pron(interrog(O,how_many(G-D-X,O),K-(G-N)-D-X)) -->
                    int_art(G).

pron(rel,K-A-typ(V)-X) --> r_i_pron(K).
pron(rel,K-A-D-X) --> rel_pron(K,A).


rel_pron(dir,_) --> pron2_r_i.
rel_pron(sub,_) --> pron2_r_i.

rel_pron(prep(de),A) --> pron_r(A).
rel_pron(prep(em),_) --> pron_r_i; conj_c.
rel_pron(prep(P),G-N) --> ( c,[P], {prep(P)},
                          def_art(G-N), (pronl_r_i(N);
                          pron2_r_i );
                          [P], {prep(P)}, (pron2_r_i ;
                                           pron3_r_i)).


r_i_pron(sub) --> pron3_r_i.
r_i_pron(prep(P)) --> [P], {prep(P)}, pron3_r_i.

int_pron(sub) --> pron2_r_i ; pron3_r_i.
int_pron(dir) --> pron2_r_i.
int_pron(prep(P)) --> [P], {prep(P)}, (pron2_r_i ;
                                       pron3_r_i ).
```

```
int_pron(dir) --> pronl_i.
int_pron(prep(em)) --> pron_r_i ; conj_c ; adv_m_i.


/*   ellipsis or generation of a verb */

int_prons,[ve(ser,N)] --> (pron3_r_i ; pronl_r_i(N)),
                           ve(ser,N),!.
int_prons,[ve(ter,N)] --> (pron3_r_i ; pronl_r_i(N)),
                          ve(ter,N),!.
int_prons,[ve(estar,N)] --> adv_m_i,ve(estar,N),!.
int_prons,[ve(ser,N)] --> (pron3_r_i ; pronl_r_i(N)).


int_pronsl,[ve(ser,N)] --> pron3_r_i, ve(ser,N).
int_pronsl,[ve(ter,N)] --> pron3_r_i, ve(ter,N).


/* Adverbs */

adver, [def_art(_)] --> adv_m_i.


/* Separators: conjunction, disjunction and comma */

com,[end] --> [].
com,[com(C)] --> [].


end(e),[end,X] --> [X],({X=('?');X='!';X='.'}).
end(comma),[com(comma)] --> [','].
end(comma),[com(e)] --> e.

end(disj),[com(ou)] --> ou.

end(e),[end] --> comma.
end(ou),[end] --> [].

comma --> [','],!.
comma --> [].

e --> [e],!.
e --> [].

ou --> [ou].


/* Interrogative articles */

interrog_art(K-(G-N)-X,O,which((G-N)-X,K,O))-->
                          pron2_r_i ; pronl_r_i(N).
```

```
interrog_art(K-(G-plu)-D-X,O,how_many([G-X,D],O)) -->
                                int_art(G).


/* Articles and classical quantifiers */

art(X,O1,O2,O) --> [art(X,O1,O2,O)].

art((G-sin)-D-X,O1,O2,for([X,D],O1,
          if(card(X,equal,1),O2))) --> def_art(G-sin).
art((G-plu)-D-X,O1,O2,for([X,D],O1,
          if(card(X,greater,0),O2))) --> def_art(G-plu).
art(A-D-X,O1,O2,for([X,D],O1,
          if(card(X,equal,I),O2))) --> def_art(A),nb(I).
art((G-sin)-D-X,O1,O2,for([X,D],and(O1,O2),
          card(X,greater,0))) --> ind_art(G-sin).
art((G-plu)-D-X,O1,O2,for([X,D],and(O1,O2),
          card(X,greater,1))) --> ind_art(G-plu).
art((G-plu)-D-X,O1,O2,for([X,D],and(O1,O2),
          card(X,equal,Y))) --> nb(Y).
art(A-D-X,O1,O2,for([X,D],and(O1,not(O2)),
          card(X,equal,0))) --> todo_art(A).
art(A-D-X,O1,O2,for([X,D],and(O1,not(O2)),
          card(X,greater,0))) --> [not_all(A)],{negative}.
art((G-sin)-D-X,O1,O2,not(for([X,D],and(O1,O2),
          card(X,greater,0)))) --> none(G-sin),{negative}.
art((G-plu)-D-X,O1,O2,not(for([X,D],and(O1,O2),
          card(X,greater,1)))) --> none(G-plu),{negative}.



/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                     CORE VOCABULARY                          */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

/* Fixed morphology: gender and number   */
/* total of words:   101    */


/* Articles */

articles --> def_art(_);ind_art(_);nonel;none(_);
             todo_art(_);not_all(_);int_art(_).


/* Definite articles */

def_art(mas-sin) --> [o].
def_art(fem-sin) --> [a].
def_art(mas-plu) --> [os].
def_art(fem-plu) --> [as].

def_art(A) --> [def_art(A)].
```

```
/* Indefinite articles and pronouns */

ind_art(_), [pessoa] --> [alguem].

ind_art(mas-sin) --> [um];[algum].
ind_art(fem-sin) --> [uma];[alguma].
ind_art(mas-plu) --> [uns];[alguns].
ind_art(fem-plu) --> [umas];[algumas].

ind_art(A) --> [ind_art(A)].


none(_), [pessoa] --> nonel.
none(_), [tipo] --> nonel.

none(mas-sin) --> [nenhum].
none(fem-sin) --> [nenhuma].
none(mas-plu) --> [nenhuns].
none(fem-plu) --> [nenhumas].
none(A) --> [none(A)].


nonel --> [ninguem].


not_all(A) --> [nem], todo_art(A).


todo_art(mas-sin) --> [todo,o].
todo_art(fem-sin) --> [toda,a].
todo_art(_-sin) --> [qualquer].
todo_art(mas-plu) --> [todos,os].
todo_art(fem-plu) --> [todas,as].
todo_art(_-plu) --> [quaisquer].

todo_art(G-sin) --> [cada].

todo_art(A) --> [todo_art(A)].



/* Interrogative articles */

int_art(mas) --> [quantos].
int_art(fem) --> [quantas].



/* Pronouns */

pronouns --> per_pron;pron_r_i;pronl_r_i(_);pron2_r_i;
             pron3_r_i;pron_r(_);pronl_i.
```

```
/* Personal pronouns */

per_pron --> [me].


/* Relative and interrogative pronouns */

pron_r_i --> [onde].

pronl_r_i(sin) --> [qual].
pronl_r_i(plu) --> [quais].

pron2_r_i --> [que];[o,que].

pron3_r_i --> [quem].


/* Relative pronouns */

pron_r(mas-sin) --> [cujo];[o,qual].
pron_r(fem-sin) --> [cuja];[a,qual].
pron_r(mas-plu) --> [cujos];[os,quais].
pron_r(fem-plu) --> [cujas];[as,quais].


/* Interrogative pronouns */

pronl_i --> [quanto].


/* Adverbs */

adverbs --> adv_m_i;adv_m;adv_q.

adv_m_i --> [como].
adv_m --> [bem].
adv_q --> [mais].
```

```
/* Prepositions */

prep(a).
prep(apos).
prep('ate''').
prep(com).
prep(de).
prep(desde).
prep(em).
prep(entre).
prep(para).
prep(por).
prep(sobre).


/* Conjunctions */


conj_c --> [quando].


/* Contraction of prepositions */


contract,[a,a] --> [''a'].
contract,[a,o] --> [ao].
contract,[a,os] --> [aos].

contract,[de,o] --> [do].
contract,[de,os] --> [dos].
contract,[de,a] --> [da].
contract,[de,as] --> [das].

contract,[em,o] --> [no].
contract,[em,os] --> [nos].
contract,[em,a] --> [na].
contract,[em,as] --> [nas].

contract,[em,um] --> [num].
contract,[em,uma] --> [numa].
contract,[em,uns] --> [nuns].
contract,[em,umas] --> [numas].

contract,[por,o] --> [pelo].
contract,[por,a] --> [pela].
contract,[por,os] --> [pelos].
contract,[por,as] --> [pelas].


contract --> [].
```

```
/* Numerals */

nb(I) --> [I],{integer(I)}.

nb(1) --> [um];[uma].
nb(2) --> [dois].
nb(3) --> [tres].
nb(4) --> [quatro].
nb(5) --> [cinco].
nb(6) --> [seis].
nb(7) --> [sete].
nb(8) --> [oito].
nb(9) --> [nove].
nb(10) --> [dez].


/* Pre-locutory expressions */

expressions --> expl;exp2.


expl --> [por,favor];['sera'',que];[diga,me,se];
         [gostaria,de,saber,se].
exp2 --> ( [de] ; [diga] ), per_pron.


pre_loc1 --> expl, { +question}.
pre_loc2, [qual] --> exp2, { +diga}.



:-end.




dictio.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                  PERIPHERAL VOCABULARY                       */
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/



/* Non-fixed morphology for the library world      */
/* total of words:   188(single) ; 113(compound)   */
```

```
/* Common nouns: definitions */

noun([A-typ(aut([]))-X],pr(author(X))) --> no(aut,A).
noun([A-typ(aut([]))-X,prep(de)-_-tit(V)-Y],
                    pr(author(X,Y))) --> no(aut,A).
noun([A-typ(pub([]))-X],pr(publisher(X))) --> no(edi,A).
noun([A-typ(pub([]))-X,prep(de)-_-tit(V)-Y],
                    pr(pub_of_tit(X,Y))) --> no(edi,A).

noun([A-tit(V)-X],pr(title(X))) --> no(tit,A).
noun([A-tit(V)-X,prep(de)-_-typ(aut([]))-Y],
                    pr(author(Y,X))) --> no(tit,A).
noun([A-tit(V)-X,prep(de)-_-typ(pub([]))-Y],
                    pr(pub_of_tit(Y,X))) --> no(tit,A).

noun([A-yea([])-X],pr(year(X))) --> no(ano,A).
noun([A-yea([])-Y,prep(de)-_-tit(V)-X],
                    pr(year_of(X,Y))) --> no(ano,A).

noun([A-tit(V)-X],pr(class(X))) --> no(doc,A).

noun([A-W-X,prep(de)-_-tit(V)-Y],
                    pr(class(X,Y))) --> no(cla,A).

noun([A-cat([])-X],pr(category(X))) --> no(cat,A).
noun([A-_-_,prep(para)-_-cat([])-X],
                    pr(category(X))) --> no(cat,A).
noun([A-cat([])-X,prep('ate'')-_-cat([])-Y],
                    pr(upon(X,Y))) --> no(cat,A).

noun([A-cat([])-X,prep(de)-_-tit(V)-Y],
                    pr(category(X,Y))) --> no(cat,A).
noun([A-cat([])-X,prep(para)-_-tit(V)-Y],
                    pr(category(X,Y))) --> no(cat,A).

noun([A-ref([])-X,prep(de)-_-tit(V)-Y],
                    pr(reference(X,Y))) --> no(ref,A).
noun([A-ref([])-X,prep(sobre)-_-tit(V)-Y],
                    pr(reference(X,Y))) --> no(ref,A).

noun([A-ref([])-X,prep(para)-_-tit(V)-Y],
                    pr(reference(X,Y))) --> no(ref,A).

noun([A-tit(boo([]))-X],pr(tit_of_book(X))) --> no(liv,A).
noun([A-tit(art([]))-X],pr(tit_of_paper(X))) -->
                                            no(art,A).

noun([A-tit(boo([]))-X,prep(sobre)-_-cat([])-Y],
                    pr(book_cat(X,Y))) --> no(liv,A).
noun([A-tit(art([]))-X,prep(sobre)-_-cat([])-Y],
                    pr(paper_cat(X,Y))) --> no(art,A).

noun([A-tit(boo([]))-X,prep(de)-_-typ(aut([]))-Y],
                    pr(aut_of_book(Y,X))) --> no(liv,A).
```

```
noun([A-tit(boo([]))-X,prep(de)-_-typ(pub([]))-Y],
                    pr(pub_of_book(Y,X))) --> no(liv,A).

noun([A-tit(art([]))-X,prep(de)-_-typ(aut([]))-Y],
                    pr(aut_of_paper(Y,X))) --> no(art,A).
noun([A-tit(aut([]))-X,prep(de)-_-typ(pub([]))-Y],
                    pr(pub_of_paper(Y,X))) --> no(art,A).

noun([A-cat([])-X],pr(cat_label(X,Y))) --> no(nom,A).

noun([A-cat([])-X,prep(de)-_-_-Y],
                    pr(cat_label(X,Y))) --> no(nom,A).

noun([A-doc(V)-X,prep(sobre)-_-tit(V)-Y],
                    pr(info(X,Y))) --> no(inf,A).


/* List of common nouns */

no(Type,GN) --> [Name], { nol(Name,Type,GN) }.

nol(nome,nom,mas-sin).

nol(dialogo,_,mas-sin).
nol(conversa,_,fem-sin).

nol(masculino,_,mas-sin).
nol(masculina,_,fem-sin).
nol(feminino,_,mas-sin).
nol(feminina,_,fem-sin).

nol(documento,doc,mas-sin).
nol(publicacao,doc,fem-sin).
nol(documentos,doc,mas-plu).
nol(publicacoes,doc,fem-plu).

nol(autor,aut,mas-sin).
nol(escritor,aut,mas-sin).
nol(tipo,aut,mas-sin).
nol(editor,aut,mas-sin).
nol(pessoa,aut,fem-sin).
nol(autores,aut,mas-plu).
nol(escritores,aut,mas-plu).
nol(tipos,aut,mas-plu).
nol(editores,aut,mas-plu).
nol(pessoas,aut,fem-plu).

nol(livro,liv,mas-sin).
nol(livros,liv,mas-plu).

nol(artigo,art,mas-sin).
nol(relatorio,art,mas-sin).
nol(artigos,art,mas-plu).
nol(relatorios,art,mas-plu).
```

```
nol(livreiro,edi,mas-sin).
nol(livreira,edi,fem-sin).
nol(livreiros,edi,mas-plu).
nol(livreiras,edi,fem-plu).

nol(classe,cla,fem-sin).
nol(especie,cla,fem-sin).

nol(titulo,tit,mas-sin).
nol(titulos,tit,mas-plu).

nol(ano,ano,mas-sin).
nol(data,ano,fem-sin).
nol(anos,ano,mas-plu).
nol(datas,ano,fem-plu).

nol(classificacao,cat,fem-sin).
nol(categoria,cat,fem-sin).
nol(classificacoes,cat,fem-plu).
nol(categorias,cat,fem-plu).

nol(referencia,ref,fem-sin).
nol(bibliografia,ref,fem-sin).
nol(referencias,ref,fem-plu).

nol(informacao,inf,fem-sin).
nol(informacoes,inf,fem-plu).


/* Single and compound proper nouns */


/*   gender of proper nouns */

gen(masculino,mas).
gen(masculina,mas).
gen(feminino,fem).
gen(feminina,fem).


/*   domain of proper nouns */

dom(autor,typ(V)).
dom(categoria,cat([])).
dom(livreiro,typ(pub(V))).
dom(titulo,tit(V)).


/* list of single proper nouns */


pn(allwood,mas,typ(V)).
```

```
pn(back,mas,typ(V)).
pn(bobrow,mas,typ(V)).
pn(brown,mas,typ(V)).
pn(bundy,mas,typ(V)).
pn(burstall,mas,typ(V)).
pn(charniak,mas,typ(V)).
pn(coelho,mas,typ(V)).
pn(colmerauer,mas,typ(V)).
pn(eder,mas,typ(V)).
pn(fikes,mas,typ(V)).
pn(fillmore,mas,typ(V)).
pn(hewitt,mas,typ(V)).
pn(kowalski,mas,typ(V)).
pn(lehnert,fem,typ(V)).
pn(loveland,mas,typ(V)).
pn(mcdermott,mas,typ(V)).
pn(mendelson,mas,typ(V)).
pn(minsky,mas,typ(V)).
pn(newell,mas,typ(V)).
pn(nilsson,mas,typ(V)).
pn(reboh,mas,typ(V)).
pn(robison,mas,typ(V)).
pn(rulifson,mas,typ(V)).
pn(sacerdoti,mas,typ(V)).
pn(schank,mas,typ(V)).
pn(shortliffe,mas,typ(V)).
pn(simmons,mas,typ(V)).
pn(slagle,mas,typ(V)).
pn(sussman,mas,typ(V)).
pn(tarski,mas,typ(V)).
pn(tate,mas,typ(V)).
pn(warren,mas,typ(V)).
pn(weizenbaum,mas,typ(V)).
pn(wilks,mas,typ(V)).
pn(winograd,mas,typ(V)).
pn(winston,mas,typ(V)).

/* List of compound proper nouns */

pn('psychology of computer vision',fem,tit(V)).
pn('problem solving methods in artificial intelligence',
   mas,tit(V)).
pn('understanding natural language',fem,tit(V)).
pn('computer models of thought and language',mas,tit(V)).
pn('learning structural descriptions from examples',
   fem,tit(V)).
pn('a computational model of skill and acquisition',
   mas,tit(V)).
pn('human problem solving',fem,tit(V)).
pn('universals in linguistic theory',mas,tit(V)).
pn('toward a model of children''s story comprehension',
   mas,tit(V)).
```

```
pn('computer power and human reason',fem,tit(V)).
pn('scripts,plans,goals and understanding',mas,tit(V)).
pn('artificial intelligence',fem,tit(V)).
pn('logic in linguistics',fem,tit(V)).
pn('introduction to logic',fem,tit(V)).
pn('introduction to mathematical logic',fem,tit(V)).
pn('automated theorem proving:a logical basis',
   fem,tit(V)).
pn('computer-based medical consultations:mycin',
   fem,tit(V)).
pn('programming in pop-2',fem,tit(V)).
pn('ai:the heuristic programming approach',fem,tit(V)).
pn('the process of question answering',mas,tit(V)).
pn('computational semantics',fem,tit(V)).


pn('a framework for representing knowledge',fem,tit(V)).
pn('description and theoretical analysis (using schemata)
   of planner',fem,tit(V)).
pn('strips:a new approach to the application of theorem
   proving to problem solving',fem,tit(V)).
pn('identification of conceptualizations underlying
   natural language',fem,tit(V)).
pn('the conniver reference manual',mas,tit(V)).
pn('semantic networks:their computation and use for
   understanding english sentences',mas,tit(V)).
pn('planning in a hierarchy of abstraction spaces',
   mas,tit(V)).
pn('margie:memory analysis response generation and
   inference on english',fem,tit(V)).
pn('qa4:a procedural calculus for intuitive reasoning',
   mas,tit(V)).
pn('a preliminary qlisp manual',mas,tit(V)).
pn('the case for case',mas,tit(V)).
pn('learning and executing generalized robot plans',
   fem,tit(V)).
pn('new programming languages for artificial intelligence
   research',fem,tit(V)).
pn('a machine oriented logic based on the resolution
   principle',fem,tit(V)).
pn('artificial intelligence and learning strategies',
   fem,tit(V)).
pn('logic for problem solving',fem,tit(V)).
pn('project planning using a hierarchic non-linear
   planner',mas,tit(V)).
pn('a prolog-like interpreter for non-linear clauses',
   mas,tit(V)).
pn('making preferences more active',fem,tit(V)).
pn('implementing prolog-compiling logic programs',
   fem,tit(V)).
pn('logic programming and compiler writing',fem,tit(V)).
pn('les grammaires de metamorphose',fem,tit(V)).
pn('planner:a language for proving theorems in robots',
   fem,tit(V)).
```

```
pn('exploiting the properties of function to control
   search',fem,tit(V)).
pn('geom:a prolog geometry theorem prover',mas,tit(V)).

pn('academic press',fem,typ(pub(V))).
pn('addison-wesley',fem,typ(pub(V))).
pn('cambridge press',fem,typ(pub(V))).
pn('edinburgh press',fem,typ(pub(V))).
pn(elsevier,fem,typ(pub(V))).
pn(freeman,mas,typ(pub(V))).
pn('holt,rinehart and winston',mas,typ(pub(V))).
pn('lawrence erlbaum',mas,typ(pub(V))).
pn('mcgraw hill',fem,typ(pub(V))).
pn('mit ai lab',mas,typ(pub(V))).
pn('north-holland',fem,typ(pub(V))).
pn('oxford press',fem,typ(pub(V))).
pn('prentice hall',fem,typ(pub(V))).
pn('project mac mit',mas,typ(pub(V))).
pn('van nostrand reinhold',fem,typ(pub(V))).
pn('yale press',fem,typ(pub(V))).


pn('acm survey',mas,typ(pub(V))).
pn(ai,mas,typ(pub(V))).
pn(dai,mas,typ(pub(V))).
pn(gia,mas,typ(pub(V))).
pn(ijcai,fem,typ(pub(V))).
pn('journal of acm',mas,typ(pub(V))).
pn(lnec,mas,typ(pub(V))).
pn(sai,fem,typ(pub(V))).
pn(sri,mas,typ(pub(V))).


pn('computer sciences',fem,cat([])).
pn('computer sciences applications',fem,cat([])).
pn(software,fem,cat([])).
pn('mathematics of computation',fem,cat([])).
pn(humanities,fem,cat([])).
pn('artificial intelligence',fem,cat([])).
pn('programming languages',fem,cat([])).
pn(processors,mas,cat([])).
pn(metatheory,fem,cat([])).
pn('language translation and linguistics',fem,cat([])).
pn('artificial intelligence applications',fem,cat([])).
pn(logic,fem,cat([])).
pn('formal languages',fem,cat([])).
pn('computational logic',fem,cat([])).
pn('mathematics,science,and engineering aids',
   fem,cat([])).
pn('authomatic theorem proving',fem,cat([])).
pn(robots,mas,cat([])).
pn('machine vision',fem,cat([])).
pn('natural language systems',mas,cat([])).
pn('information processing psychology',fem,cat([])).
```

```
pn('artificial intelligence tools',mas,cat([])).
pn('theory of heuristic methods',fem,cat([])).
pn('modelling and representation of knowledge',
   fem,cat([])).
pn('common-sense reasoning,deduction and
   problem solving',mas,cat([])).
pn('artificial intelligence systems and languages',
   fem,cat([])).
pn(compilers,mas,cat([])).
pn(interpreters,mas,cat([])).
pn('induction and hypothesis formation',fem,cat([])).
pn('learning and adaptive systems',fem,cat([])).
pn('frame systems',mas,cat([])).
pn('predicate calculus',mas,cat([])).
pn('puzzle solving',fem,cat([])).
pn('question answering',fem,cat([])).
pn('common-sense reasoning',mas,cat([])).
pn(planning,mas,cat([])).
pn('pop-2',mas,cat([])).
pn(planner,mas,cat([])).
pn(qlisp,mas,cat([])).
pn(conniver,mas,cat([])).
pn(prolog,mas,cat([])).
pn('resolution principle',mas,cat([])).
pn('ad hoc methods',mas,cat([])).


pn(I,_,_):-integer(I).


/* Verbs: definitions */

verb(L,O) --> [verb(L,O)].

verb([(G-N)-D-X,noun-A-D-Y],pr(set_equal(X,Y))) -->
                                        ve(ser,N).
verb([(G-N)-D1-X,dir-A-D2-Y],pr(have(D1,X,D2,Y))) -->
                                        ve(ter,N).

verb([A-D-X,noun-A1-D1-X],t) --> copula(A).
verb([A-X,..L],O) --> copula(A), adj([A-X,..L],O).
verb([A-D-X,Y],O) --> copula(A), noun([A-D-X,Y],O).

verb([A-D-X,noun-A1-D1-X],t),[um,titulo] -->
                                    ve(haver,sin).
verb([A-D-X,noun-A1-D1-X],t),[titulos] --> ve(haver,plu).
verb([(_-N)-D-X],t) --> ve(haver,N).
verb([(_-N)-D-X],t), [ind_art(_-N)] --> ve(haver,N).

verb([(G-N)-typ(V)-X,dir-A-tit(W)-Y],
            pr(author(X,Y))) --> ve(esc,N).
verb([(G-N)-typ(V)-X,prep(para)-_-typ(pub([]))-Y],
            pr(pub_of_aut(Y,X))) --> ve(esc,N).
```

```
verb([(G-N)-typ(V)-X,prep(para)-_-typ(pub([]))-Z,
      dir-_-tit(V1)-Y],
            pr(aut_for_pub(X,Y,Z))) --> ve(esc,N).

verb([(G-N)-D1-X,dir-A-D2-Y],
            pr(publisher(D1,X,D2,Y))) --> ve(pub,N).

verb([(G-N)-typ(aut([]))-X,prep(em)-_-typ(pub([]))-Y],
            pr(pub_of_aut(Y,X))) --> ve(pub,N).

verb(L,O) --> ve(aux,sin), verb(L,O).

verb([dir-A-tit(V)-Y],pr(category(X,Y))) -->
                                ve(cat,imp,sin) ;
                                ve(cat,inf,_).
verb([(G-N)-cat([])-X,dir-A-tit(V)-Y],
            pr(category(X,Y))) --> ve(cat,pres,N).
verb([(G-N)-tit(V)-X,prep(em)-A-cat([])-Y],
            pr(category(Y,X))) --> ve(cat,pres,N).

verb([dir-A-cat([])-X],pr(category(X))) -->
                                ve(cri,imp,sin);
                                ve(cri,inf,_).

verb([dir-A-V-X],pr(destroy(X))) --> ve(apa,imp,sin);
                                     ve(apa,inf,_).

verb([dir-A-tit(V)-X],pr(archive(X))) --> ve(arq,imp,sin);
                                          ve(arq,inf,_).

verb([prep(sobre)-A-tit(V)-Y],pr(info(X,Y))) -->
                                   ve(inf,imp,_).

verb([(G-N)-cat([])-X,prep(de)-_-cat([])-Y],
            pr(under(X,Y))) --> ve(dep,N).

verb([(G-N)-cat([])-X,dir-_-cat([])-Y],
            pr(upon(X,Y))) --> ve(dom,N).

verb([(G-N)-cat([])-X,dir-_-cat([])-Y],
            pr(cat_label(X,Y))) --> ve(sig,N).

verb([(G-N)-cat([])-X,prep(´`a´)-_-cat([])-Y],
            pr(cat_label(X,Y))) --> ve(atr,N).


copula(_-N) --> ve(ser,N).
copula(_-N) --> ve(estar,N).


/* List of verbs */

ve(V,N) --> [ve(V,N)].
```

```
ve(Type,N) --> [Verb], {vel(Verb,Type,N)}.
ve(Type,N) --> [Verb1,Verb2], {vel(Verb1-Verb2,Type,N)}.
ve(Type,Time,N) --> [Verb], {vel(Verb,Type,Time,N)}.

vel('e´´´',ser,sin).
vel(foi,ser,sin).
vel(sao,ser,plu).
vel(foram,ser,plu).

vel('esta´´´',estar,sin).
vel(estao,estar,plu).

vel('ha´´´',haver,sin).
vel(existe,haver,sin).
vel(conhece,haver,sin).
vel(existem,haver,plu).

vel(tem,ter,_).
vel(tenho,ter,sin).

vel(escreve,esc,sin).
vel(escreveu,esc,sin).
vel(escrevem,esc,plu).
vel(escreveram,esc,plu).

vel(publica,pub,sin).
vel(publicou,pub,sin).
vel(publicam,pub,plu).
vel(publicaram,pub,plu).

vel(depende,dep,sin).
vel(dependem,dep,plu).

vel(domina,dom,sin).
vel(dominam,dom,plu).

vel(significa,sig,sin).
vel(diz,sig,sin).
vel(quer-dizer,sig,sin).

vel(atribui,atr,sin).
vel(chama,atr,sin).

vel(quero,aux,sin).
vel(conheco,aux,sin).

vel(concordo,afir,sin).
vel(V,afir,sin):-vel(V,aux,sin);
                 vel(V,ter,_);
                 vel(V,estar,_).

vel(desisto,des,_).
vel(deixe,des,_).
vel(mude,des,_).
```

```
vel(sei,sei,sin).

vel(arquive,arq,imp,sin).
vel(arquivar,arq,inf,_).

vel(apague,apa,imp,sin).
vel(destrua,apa,imp,sin).
vel(esqueca,apa,imp,sin).
vel(apagar,apa,inf,_).
vel(destruir,apa,inf,_).
vel(esquecer,apa,inf,_).

vel(classificar,cat,inf,_).
vel(classifique,cat,imp,sin).
vel(classifica,cat,pres,sin).

vel(crie,cri,imp,sin).
vel(criar,cri,inf,_).

vel(informe,inf,imp,_).


/* Adjectivs: definitions */

adj([A-tit(V)-X,prep(em)-_-typ(pub([]))-Y],
                        pr(pub_of_tit(Y,X))) --> ad(edi,A).

adj([A-typ(aut([]))-X,prep(por)-_-typ(pub([]))-Y],
                        pr(pub_of_aut(Y,X))) --> ad(edi,A).
adj([A-tit(V)-X,prep(por)-_-typ(pub([]))-Y],
                        pr(pub_of_tit(Y,X))) --> ad(edi,A).

adj([A-tit(V)-X],pr(title(X))) --> ad(pub,A).
adj([A-typ(aut([]))-X,prep(em)-_-yea([])-Y],
                        pr(year_aut(X,Y))) --> ad(pub,A).
adj([A-tit(V)-X,prep(em)-_-yea([])-Y],
                        pr(year_of(X,Y))) --> ad(pub,A).

adj([A-tit(V)-X,prep(em)-_-yea([])-Z,
     prep(por)-_-typ(pub([]))-Y],
             pr(year_tit_pub(Z,X,Y))) --> ad(pub,A).

adj([A-tit(V)-X,prep(apos)-_-yea([])-Y],
             pr(published_after(X,Y))) --> ad(pub,A).
adj([A-tit(V)-X,prep(apos)-_-yea([])-Y],
             pr(published_after(X,Y))) --> ad(pub,A).
adj([A-tit(V)-X,prep(desde)-_-yea([])-Y],
             pr(published_from(X,Y))) --> ad(pub,A).

adj([A-tit(V)-X,prep(e)-_-yea([])-Z,
     prep(entre)-_-yea([])-Y],
             pr(published_between(X,Y,Z))) --> ad(pub,A).
```

```
adj([A-tit(V)-X,prep(por)-_-typ(aut([]))-Y],
                    pr(author(Y,X))) --> ad(esc,A).

adj([A-cat(V)-X,dir-_-tit(Vl)-Y],
                    pr(category(X,Y))) --> ad(cat,A).

adj([A-tit(boo([]))-X,prep(em)-_-cat(V)-Y],
                    pr(book_cat(X,Y))) --> ad(cat,A).

adj([A-tit(art([]))-X,prep(em)-_-cat(V)-Y],
                    pr(paper_cat(X,Y))) --> ad(cat,A).

adj([A-tit(boo([]))-X,prep(com)-_-tit(V)-Y],
                    pr(book_ref(X,Y))) --> ad(rel,A).

adj([A-tit(art([]))-X,prep(com)-_-tit(V)-Y],
                    pr(paper_ref(X,Y))) --> ad(rel,A).

adj([A-tit(boo([]))-X,prep(com)-_-tit(V)-Y],
                    pr(ref_book(X,Y))) --> ad(ref,A).

adj([A-tit(art([]))-X,prep(com)-_-tit(V)-Y],
                    pr(ref_paper(X,Y))) --> ad(ref,A).

adj([A-tit(boo([]))-X,prep(em)-_-tit(V)-Y],
                    pr(book_ref(X,Y))) --> ad(ref,A).

adj([A-tit(art([]))-X,prep(em)-_-tit(V)-Y],
                    pr(paper_ref(X,Y))) --> ad(ref,A).

adj([A-cat([])-X,_-_-cat([])-Y],
                    pr(link(X,Y))) --> ad(lig,A).
adj([A-cat([])-X,prep(´`a´)-_-cat([])-Y],
                    pr(link(X,Y))) --> ad(lig,A).
adj([A-cat([])-X,prep(com)-_-cat([])-Y],
                    pr(link(X,Y))) --> ad(lig,A).

adj([A-cat([])-X,prep(de)-_-cat([])-Y],
                    pr(under(X,Y))) --> ad(dep,A).

adj([A-cat([])-X,prep(de)-_-cat([])-Y],
                    pr(upon(X,Y))) --> ad(dom,A).

adj([A-cat([])-X,prep(´`a´)-_-cat([])-Y],
                    pr(near(X,Y))) --> ad(jun,A).

adj([A-tit(V)-X],pr(reference(X,[]))) --> ad(ato,A).


/* List of adjectives */

ad(Type,GN) --> [Adj], { adl(Adj,Type,GN) }.
```

```
adl(autonomo,ato,mas-sin).
adl(autonomo,ato,mas-plu).

adl(editado,edi,mas-sin).
adl(editada,edi,fem-sin).
adl(editados,edi,mas-plu).
adl(editadas,edi,fem-plu).

adl(publicado,pub,mas-sin).
adl(chamado,pub,mas-sin).
adl(publicada,pub,fem-sin).
adl(publicados,pub,mas-plu).
adl(arquivados,pub,mas-plu).
adl(publicadas,pub,fem-plu).

adl(escrito,esc,mas-sin).
adl(escritos,esc,mas-plu).

adl(conhecido,cat,mas-sin).
adl(classificado,cat,mas-sin).
adl(classificada,cat,fem-sin).
adl(classificados,cat,mas-plu).
adl(classificadas,cat,fem-plu).

adl(referenciado,ref,mas-sin).
adl(referenciada,ref,fem-sin).
adl(referenciados,ref,mas-plu).
adl(referenciadas,ref,fem-plu).

adl(relacionado,rel,mas-sin).
adl(relacionada,rel,fem-sin).
adl(relacionados,rel,mas-plu).
adl(relacionadas,rel,fem-plu).

adl(ligada,lig,fem-sin).
adl(ligadas,lig,fem-plu).

adl(dependente,dep,_-sin).
adl(abaixo,dep,_).
adl(dependentes,dep,_-plu).

adl(dominante,dom,_-sin).
adl(acima,dom,_).
adl(dominantes,dom,_-plu).

adl(junto,jun,_-sin).



:-end.
```

```
dbms.

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*              DATA BASE MANAGEMENT SYSTEM                      */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/


/* Access to the data base */

/* search of all individuals */

find_all(_,_,_,_):- +elem('&top',0),fail.
find_all([X,D],not(O),_,_):-
                              !,find_all_non(X,D,not(O)).
find_all([[X],_],O1,O2,B):-verify(O1,v),elem2(X,N),
                  incompatible(O2,N,B),!, -all(elem(_,_)).
find_all([X,_],_,O2,B):-calcul(X,[]),!,
                         (compatible(X,O2,B),!;B=f).


/*     for cases where negation is involved  */

find_all_non([X],D,not(O)):-domain(_,D,X),
                             verify(not(O),v),
                             +elem(X,_),fail.
find_all_non(X,_,_):-calcul(X,[]),!.


/*     calcul of all found individuals */

calcul(X,Y):- -elem(A,_),
              ( A='&top',X=Y,! ;
               member(A,Y),calcul(X,Y),! ;
               calcul(X,[A,..Y]) ).


/* compatibility tests */

incompatible(if(O,_),N,u):-incompatible(O,N,_),!.

incompatible(card(_,_,X),_,f):-var(X),!,fail.
incompatible(card(_,equal,0),N,f):-N==0.
incompatible(card(_,equal,1),N,f):-N==1.


compatible(X,if(O1,O2),B):-presuppose(X,O1,O2,B).

compatible([],card(_,equal,0),t).
```

```
compatible([X],card(_,equal,1),t).
compatible([X|L],card(_,greater,0),t).
compatible(X,card(_,equal,J),t):-length(X,J).


elem2(X,N):- ?elem(_,M),!,N is M+1, +elem(X,N).


/* getting individuals satisfying a relation */

pick_up(A,R):- P=..[R,A], (P;domain(R,_,A)).


pick_up(A,B,i(R)):-!,pick_up(B,A,R).
pick_up(A,B,R):- P=..[R,A,B], P.


pick_up(A,B,C,R):- P=..[R,A,B,C], P.



/* Primitives */

/* numeral of a set */

card(X,equal,Y):-card(X,Y).
card(X,greater,Y):-card(X,Z), Y<Z.


card([],0).
card([A,..L],C):-card(L,C1), C is C1+1.



/* operations upon sets */

member(A,[A,.._]).
member(A,[_,..L]):-member(A,L).


equal_lists([],[]).
equal_lists([X,..L1],L2):- delete(X,L2,L3),
                           equal_lists(L1,L3).


delete(X,[X,..Y],Y).
delete(X,[Y,..L1],[Y,..L2]):- delete(X,L1,L2).


get_till_n([],[],_,_).
get_till_n([X,..L],[X,..P],C,N):-(var(C),C=1;true),
                                  C=<N,C1 is C+1,
                                  get_till_n(L,P,C1,N).
get_till_n([],_,_,_).
```

```
get_till_5([],[],_,C).
get_till_5([X,..P],[X,..L],R,C):-(var(C),C=1;true),
                                  C<6, C1 IS C+1,
                                  get_till_5(P,L,R,C1).
get_till_5([],R,R,_).


compact([L,[]],L).
compact([L],L).
compact([L1,[L2]],L):-concatenate(L1,L2,L).
compact([L1,L2,..LN],L):-concatenate(L1,L2,X),
                         compact([X,LN],L).


concatenate([X,..L1],L2,[X,..L3]):-concatenate(L1,L2,L3).
concatenate([],L,L).


simplify(L,NL):-compact(L,L1),simplify1(L1,L2),
                simplify2(L2,NL).

simplify1([X,..L],NL):-member(X,L),!,simplify1(L,NL).
simplify1([X,..L],[X,..NL]):-simplify1(L,NL).
simplify1([],[]).


simplify2(L,NL):-simplify3(L,L1),subtract(L,L1,NL).


simplify3([X,..L],NL):-compare(L,X,[]),simplify3(L,NL).
simplify3([X,..L],[Y,..NL]):-compare(L,X,Y),
                             simplify3(L,NL).
simplify3([],[]).


compare(_,[],[]):-!.
compare([Y,..L],X,[Z,..NL]):-(linked(X,Y),Z=X;
                              linked(Y,X),Z=Y),
                              compare(L,X,NL).
compare([Y,..L],X,NL):-compare(L,X,NL).
compare([],_,[]).


subtract(L,[],L):-!.
subtract([H,..T],L,U):-member(H,L),!,subtract(T,L,U).
subtract([H,..T],L,[H,..U]):-!,subtract(T,L,U).
subtract(_,_,[]).


/* logical operations */

and(_,u,u):-!.
and(u,_,u):-!.
and(t,and(t,t),t):-!.
```

```
and(t,t,t):-!.
and(f,t,f):-!.
and(t,f,f):-!.
and(O1,_,u):-verify(O1,u),!.
and(_,O2,u):-verify(O2,u),!.
and(O1,O2,t):-verify(O1,t),verify(O2,t).
and(_,_,f).


presuppose(X,O1,O2,B):-compatible(X,O1,_),!,
                        verify(O2,B).
presuppose(_,_,_,u).


if(O1,O2,B):-verify(O1,t),!,verify(O2,B).
if(_,_,u).


set_equal(X,X):-!.
set_equal(X,Y):- equal_lists(X,Y).

t.

not(f,t).        not(t,f).          not(u,u).


ifnot(not(_)):-fail.
ifyes(B):-var(B);B=t.


/* Verification routines */


/* logical verifications */


verify_d(X,and(B1,B2),B):-look(B1),and(B2,B1,B),!;
                          and(B1,B2,B).


look(pr(P)):-P=..[_,X,Y],var(X),var(Y).
look(and(_,and(_,_))):-fail.

look(and(X,Y)):-look(X);look(Y).


verify(t,t):-!.


/*    for singular treatment */
```

```
verify(for([X,D],O1,if(card(X,equal,1),O2)),B):-
                    and(O1,if(card(X,equal,1),O2),B).
verify(for([X,D],and(O1,O2),card(X,greater,0)),B):-
                    ifnot(O2),!,
                    and(and(O1,O2),card(X,greater,0),B).


/*    for plural treatment */

verify(for(X,O1,O2),B):-!,find_all(X,O1,O2,B).


/*    for the treatment of general cases */

verify(not(O),B2):-!,verify(O,B1),!,not(B1,B2).
verify(if(O1,O2),B):-!,if(O1,O2,B).
verify(and(O1,O2),B):-!,and(O1,O2,B).
verify(pr(O),B):-!,pr(O,B).
verify(O,t):-O,!.
verify(_,f).


/* verification of relations */


pr(P,B):- P=..[R,X],(var(X),X=[A];X=coord(disj([A]));
          X=coord([A]);
          X=disj([A]);X=[A]),!,(pick_up(A,R),B=t;B=f).

pr(P,B):- P=..[R,L1,L2],!,
        (nonvar(L1),L1=coord(disj(X)),L3=disj(X),L4=L2 ;
         L3=L1,L4=L2 ),!,
         prl(R,L3,L4,B).

pr(P,B):- P=..[R,L1,L2,L3],!,prl(R,L1,L2,L3,B).

pr(P,B):-definition(P,NP),pr(NP,B).


prl(_,_,X,u):-nonvar(X),X=[],!.
prl(_,X,_,u):-nonvar(X),X=[],!.
prl(set_equal,X,X,t):-!.
prl(set_equal,L,X,t):- equal_lists(X,L).

prl(R,[E],L,B):-var(L),!,relate_with(E,L,R,B);
                L=coord(L1),!,relate_with(E,L1,R,B);
                L=disj(L1),relate_or(E,L1,R,B),!.

prl(R,[H,..T],[E],B):-nonvar(H),nonvar(E),!,
                      relate_with(E,[H,..T],i(R),B).
prl(R,L1,[E],B):-var(L1),!,relate(L1,[E],R,B);
                 L1=disj(L),relate_or(E,L,i(R),B),!;
```

```
                    L1=coord(L),!,relate_in(E,L,i(R),B).
prl(R,L1,L2,B):-relate(L1,L2,R,B).

prl(R,[L1],[L2],[L3],B):-relate_with(L1,L2,L3,R,B).


relate([E],L,R,B):-!,relate_in(E,L,R,B).
relate([E1,..E],L,R,B):-!,relate_in(E1,L,R,B1),
                          relate(E,L,R,B2),and(B1,B2,B).


relate_with(E,[E1],R,t):-var(E),var(E1),!,
                           pick_up(E,E1,R);
                           pick_up(E,E1,R).
relate_with(E,[E1,E2,..L],R,B):-ifyes(B),
               pick_up(E,E1,R),pick_up(E,E2,R),
               ( L=[], B=t ,!; relate_with(E,L,R,B) ).
relate_with(_,_,_,f).

relate_with(L1,L2,L3,R,B):-pick_up(L1,L2,L3,R).
relate_with(_,_,_,_,f).


relate_in(E,L,R,t):-member(E1,L),pick_up(E,E1,R).
relate_in(_,_,_,f).


relate_or(E,[E1,..L],R,t):-( pick_up(E,E1,R),!;
                             relate_or(E,L,R,t)).
relate_or(_,_,_,f).


/* Routines for the archive of new facts */

archive_cat(X,Y,Z,W):-asserta(cat(X,Y,Z,W)).


archive_doc(livro,X,Y,Z,T,U,V,W):-
                           asserta(book(X,Y,Z,T,U,V,W)).
archive_doc(artigo,X,Y,Z,T,U,V,W):-
                           asserta(paper(X,Y,Z,T,U,V,W)).


archive_pn(N,G,D):-asserta(pn(N,G,D)).


erase_cat(X):-integer(X),retract(cat(_,_,_,X));
              retract(cat(_,_,X,_)).


erase_doc(X):-retract(book(_,_,X,_,_,_,_));
              retract(paper(_,_,X,_,_,_,_)).
```

```
/* Routines for the internal data base management */

+T:- recorda(T).


-all(T):- -T,fail.
-all(_).


-T:- recorded(T,P),erase(P).


:-end.




dbase.

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                        DATA BASE                          */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/



/* Declaration of the domains of the objects which
   participate in user sentences                              */

domain(doc,doc(V),X):-domain(typ,V,X);
                      domain(tit,V,X);
                      domain(yea,V,X);
                      domain(cat,V,X);
                      domain(ref,V,X).

domain(typ,typ(aut(V)),X):-domain(aut,aut(V),X).
domain(typ,typ(pub(V)),X):-domain(pub,pub(V),X).

domain(tit,tit(V),X):-domain(boo,boo(V),X);
                      domain(art,art(V),X).

domain(aut,aut([]),X):-author(X).
domain(pub,pub([]),X):-publisher(X).
domain(yea,yea([]),X):-year(X).
domain(cat,cat([]),X):-category(X).
domain(ref,ref([]),X):-reference(X,_).

domain(boo,boo([]),X):-class(livros,X).
```

```
domain(art,art([]),X):-class(artigos,X).


/* Definition of properties */

title(X):-book(_,_,X,_,_,_,_);paper(_,_,X,_,_,_,_).
tit_of_book(X):-book(_,_,X,_,_,_,_).
tit_of_paper(X):-paper(_,_,X,_,_,_,_).

author(X):-author(X,_).

publisher(X):-pub_of_tit(X,_).

year(Y):-year_of(_,Y).

class(X):-class(X,_).

category(X):-(integer(X),(cat(_,X,_,_);cat(_,_,_,X))) ;
             cat(X,_,_,_);cat(_,_,X,_).

destroy(X):-erase_cat(X);erase_doc(X).

archive(X):-title(X).

number(N):-book(_,N1,_,_,_,_,_),paper(_,N2,_,_,_,_,_),
           (N1>N2,N is N1+1;N is N2+1).


/* Definition of relations */

doc(N,T):-book(_,N,T,_,_,_,_);paper(_,N,T,_,_,_,_).

class(livros,X):-book(_,_,X,_,_,_,_).
class(artigos,X):-paper(_,_,X,_,_,_,_).

cat_label(X,Y):-cat(X,Y,_,_);cat(_,_,X,Y).

info(K-N-X-Y-Z-W-V,T):-book(X,N,T,Y,Z,W,V),K=livro,!;
                       paper(X,N,T,Y,Z,W,V),K=artigo.

category(X,Y):-var(X),
       (book(_,_,Y,_,_,X,_);paper(_,_,Y,_,_,X,_);!,fail).

book_cat(X,Y):-(integer(Y),L=Y ;
                cat(Y,L,_,_) ;
                cat(_,_,Y,L)),
               book(_,_,X,_,_,C,_),member(L,C).

paper_cat(X,Y):-(integer(Y),L=Y;
                 cat(Y,L,_,_);
                 cat(_,_,Y,L)),
                paper(_,_,X,_,_,C,_),member(L,C).
```

```
reference(X,Y):-var(X),
                (book(_,_,Y,_,_,_,X);
                 paper(_,_,Y,_,_,_,X);
                 !,fail).

book_ref(X,Y):-(integer(Y),book(_,Y,_,_,_,_,R);
                book(_,_,Y,_,_,_,R)),
               book(_,N,X,_,_,_,_),member(N,R).

ref_book(X,Y):-(integer(Y),Z=Y;book(_,Z,Y,_,_,_,_)),
                book(_,_,X,_,_,_,R),member(Z,R).

paper_ref(X,Y):-(integer(Y),paper(_,Y,_,_,_,_,R);
                 paper(_,_,Y,_,_,_,R)),
                paper(_,N,X,_,_,_,_),member(N,R).

ref_paper(X,Y):-(integer(Y),Z=Y;paper(_,Z,Y,_,_,_,_)),
                 paper(_,_,X,_,_,_,R),member(Z,R).


author(X,Y):-book(X,_,Y,_,_,_,_);paper(X,_,Y,_,_,_,_).
aut_of_book(X,Y):-book(X,_,Y,_,_,_,_).
aut_of_paper(X,Y):-paper(X,_,Y,_,_,_,_).
aut_for_pub(X,Y,Z):-book(X,_,Y,Z,_,_,_);
                    paper(X,_,Y,Z,_,_,_).

pub_of_book(X,Y):-book(_,_,Y,X,_,_,_).
pub_of_paper(X,Y):-paper(_,_,Y,X,_,_,_).

pub_of_tit(X,Y):-book(_,_,Y,X,_,_,_);paper(_,_,Y,X,_,_,_).
pub_of_aut(X,Y):-book(Y,_,_,X,_,_,_);paper(Y,_,_,X,_,_,_).

year_of(T,Y):-book(_,_,T,_,Y,_,_);paper(_,_,T,_,Y,_,_).
year_aut(X,Y):-book(_,Y,_,_,X,_,_);paper(_,Y,_,_,X,_,_).
year_tit_pub(X,Y,Z):-book(_,_,Y,Z,X,_,_);
                      paper(_,_,Y,Z,X,_,_).

published_after(T,Y1):-year_of(T,Y), Y>Y1.
published_from(T,Y1):-year_of(T,X), Y>=Y1.
published_between(T,Y,Z):-year_of(T,W), W>Y, W<Z.


definition(publisher(typ(pub([])),X,tit(V),Y),
                                    pub_of_tit(X,Y)):-!.
definition(publisher(_,X,_,Y),pub_of_tit(Y,X)).

definition(have(typ(aut([])),X,tit(V),Y),author(X,Y)):-!.
definition(have(tit(V),X,typ(aut([])),Y),author(Y,X)).

definition(have(typ(pub([])),X,typ(aut([])),Y),
                                    pub_of_aut(X,Y)):-!.
definition(have(typ(aut([])),X,typ(pub([])),Y),
                                    pub_of_aut(Y,X)).
```

```
link(X,Y):-(linked(X,Y), +lig(t);
            linked(Y,X), +lig(b)).

linked(X,Y):-cat(X,_,Y,_).
linked(X,Z):-cat(X,_,Y,_),linked(Y,Z).


above([X,..R],Y):-cat(X,_,Y,_),above(R,X).
above([],_).


down([X,..R],Y):-cat(Y,-,X,_),down(R,X).
down([],_).


upon(X,Y):-cat(X,_,Y,_).
upon(X,Z):-cat(X,_,Y,_),upon(Y,Z).


under(X,Y):-cat(Y,_,X,_).
under(X,Z):-cat(Y,_,X,_),under(Y,Z).


near(X,Y):-cat(X,_,Y,_);integer(Y),cat(X,_,_,Y).


/* File of documents: books and papers */


book(charniak,45,'computational semantics',
     'north-holland',1976,[1215,312],[3,6,9,26,28,31,43]).
book(winston,1,'psychology of computer vision',
     'mcgraw hill',1975,[1214,1216,1222],[3,4]).
book(nilsson,2,
     'problem solving methods in artificial intelligence',
     'mcgraw hill',1971,[1222,1223],[15,34]).
book(winograd,3,'understanding natural language',
     'academic press',1972,[111,1215,312],[22,31,34,43]).
book(schank,4,'computer models of thought and language',
     freeman,1973,[1215,1222],[2,3,7,9,19,31,47]).
book(winston,5,
     'learning structural descriptions from examples',
     'mit ai lab',1970,[12162],[]).
book(sussman,6,
     'a computational model of skill and acquisition',
     elsevier,1975,[12162,1222],[5,7,22]).
book(newell,7,'human problem solving','prentice hall',
     1972,[1223],[34]).
book(bach,8,'universals in linguistic theory',
     'holt,rinehart and winston',1968,[111,1215],[]).
book(charniak,9,
     'toward a model of children''s story comprehension',
     'mit ai lab',1972,[111,1215],[43]).
```

```
book(weizenbaum,10,'computer power and human reason',
     freeman,1976,[1215],[3]).
book(schank,11,'scripts,plans,goals and understanding',
     'lawrence erlbaum',1977,[1222,1223],[2,9,21,27]).
book(winston,12,'artificial intelligence',
     'addison-wesley',1977,[12],[2,6,7,10,23,25,27,31]).
book(allwool,13,'logic in linguistics','cambridge press',
     1977,[111,311],[14]).
book(tarski,14,'introduction to logic','oxford press',
     1965,[3111],[15]).
book(mendelson,15,'introduction to mathematical logic',
     'van nostrand reinhold',1964,[3111],[14]).
book(loveland,16,
     'automated theorem proving:a logical basis',
     'north-holland',1978,[1212,3111],[2,15,19,34,43]).
book(shortliffe,17,
     'computer-based medical consultations:mycin',
     elsevier,1976,[1211,1222,1223],[2,4,23,33,34]).
book(burstall,18,'programming in pop-2','edinburgh press',
     1971,[12241],[43]).
book(slagle,19,'ai:the heuristic programming approach',
     'mcgraw hill',1971,[1221],[]).
book(lehnert,20,'the process of question answering',
     'yale university',1977,[12232],[3,4,9,17,21]).


paper(coelho,46,'geom:a prolog geometry theorem prover',
      1nec,1976,[1212,12222,1223],[36]).
paper(minsky,21,'a framework for representing knowledge',
      'mit ai lab',1974,[1214,12221],[31]).
paper(hewitt,22,'description and theorectical
      analysis (using schemata) of planner',
      'project mac mit',1972,[1213,12242],[]).
paper(fikes,23,
      'strips:a new approach to the application of theorem
      proving to problem solving',ai,
      1971,[1212,1223],[2,43]).
paper(schank,24,
      'identification of conceptualization underlying
      natural language',freeman,1973,[111,1215],[]).
paper(mcdermott,25,'the conniver reference manual',
      'mit ai lab',1973,[12244],[]).
paper(simmons,26,
      'semantic networks:their computation and use for
      understanding english sentences',
      freeman,1973,[1215,12221],[31]).
paper(sacerdoti,27,
      'planning in a hierarchy of abstraction spaces',
      ijcai,1973,[12234],[22,23]).
paper(schank,28,
      'margie:memory analysis response generation and
      inference on english',ijcai,1973,
      [111,1215,12232],[3,26]).
```

```
paper(rulifson,29,
      'qa4:a procedural calculus for intuitive reasoning',
      sri,1972,[12243],[22,23,25]).
paper(reboh,30,'a preliminary qlisp manual',
      sri,1973,[12243],[29]).
paper(fillmore,31,'the case for case',
      'holt,rinehart and winston',1968,[111],[]).
paper(fikes,32,
      'learning and executing generalized robot plans',
      ai,1972,[1213,12234],[]).
paper(bobrow,33,
      'new programming languages for artificial
      intelligence research','acm survey',
      1974,[1224],[22,25,29,30]).
paper(robison,34,
      'a machine oriented logic based on the resolution
      principle','journal of acm',1965,[122331,3111],[]).
paper(brown,35,
      'artificial intelligence and learning strategies',
      bbn,1978,[12162],[]).
paper(kowalski,36,
      'logic for problem solving',sai,1974,
      [1223,3111],[2,6,22,23,29,30,34]).
paper(tate,37,
      'project planning using a hierarchic non-linear
      planner',dai,1976,[12234],[6,23,25,27,33]).
paper(eder,38,
      'a prolog-like interpreter for non-horn clauses',
      dai,1976,[12245,222,3111],[34,36]).
paper(wilks,39,
      'making preferences more active',dai,
      1977,[111,1215,312],[4,21]).
paper(warren,40,
      'implementing prolog-compiling logic programs',
      dai,1977,[12245,221,3111],[18,34,36]).
paper(warren,41,
      'logic programming and compiler writing',
      dai,1977,[221,3111],[34,36,41,42]).
paper(colmerauer,42,
      'les grammaires de metamorphose',gia,
      1975,[111,1215,312],[34]).
paper(hewitt,43,
      'planner:a language for proving theorems in robots',
      ijcai,1969,[1212,1213,12242],[]).
paper(bundy,44,
      'exploiting the properties of function
      to control search',dai,1977,[1221,1223],[23,40]).


/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
/*                      KNOWLEDGE BASE                           */
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
```

```
/* Document classification method  */

classify(N,LR,C,M):-dialogue(clas,3,N,R),
                    ?fr(u,N,aceita),N1 is N+1,
                    classification(N1,LR,C,M),
                    +cv(M),(var(C),!;test_cat(C,C1,_),
                    write('O documento ficou classificado '),
                    write('nas categorias:'),
                    nl,output(cat([]),C1)) ,nl,!.


classification(N,LR,C1,O):-
              write('Vamos classifica''-lo!'),nl,
              M is N+1,+dial(M),converse(clas-1,A,LR,LC,B),!,
              continue(LC,A,O,C1).


continue([],A,O,C1):-(var(A),!,fail;
     write('Como nao consegui obter nenhuma referencia,'),
     write(' nao sei dar-lhe qualquer sugestao!'),nl,
     N is A+1,dialogue(clas,3,N,_),N1 is N+1,
     ( ?fr(u,N,aceita), +cv(N1),
      converse(clas-4,N1-O,_,C1,_,_) ; O=N1, true),! ).

continue(LC,A,O,C1):-simplify(LC,C),method,
                      output(_,C),nl,nl,D is A+1, +cv(D),
                      converse(clas-2,D-O,_,C1,_,_).


test_cat(C,C1,C2):-accept(C,C1,C2),(C2=[],!;
   length(C2,N),(L=1,write('A categoria '),output(_,C2),
   write(' nao existe, e portanto e'' rejeitada!'),nl;
   write('As categorias '),output(_,C2),
   write(' nao existem, e portanto sao rejeitadas!'),nl)).


accept([],[],[]).
accept([X,..P],[X,..L],R):-category(X),accept(P,L,R).
accept([X,..P],L,[X,..R]):-accept(P,L,R).
accept([],_,_).


method:-write('Vamos utilizar o metodo de classificacao'),
        write(' que consiste em sugerir'),nl,
        write('como categorias para '),
        write('o documento em questao,'),nl,
        write('as obtidas das '), ?rf(K),write(K),
        write(' referencias fornecidas pelo utilizador!'),
        -rf(K),nl,
        write('Assim,proponho que o documento fique'),
        write(' classificado nas'),nl,
        write('categorias seguintes: ').
```

```
/* Archive of new documents  */

archive(N,T):-dialogue(clas,5,N,T),N1 is N+1,
              ( ?fr(u,N,aceita),number(D),
               converse(N1,Z,d(A,D,L,Y,C,R)),
               archive(Z,A,D,T,L,Y,C,R),! ;
               order(N1,clas,[T],f) ).

archive(P,T,C,R):-dialogue(arq,3,P,S),
                  ( ?fr(u,P,aceita),P1 is P+1,
                   number(M),converse(P1,d(M,A,L,Y,Z)),
                   archive(Z,A,M,T,L,Y,C,R) ; true).


archive(Z,A,M,T,L,Y,C,R):-archive_doc(Z,A,M,T,L,Y,C,R),
             write('O novo documento no.'),
             write(M),write(', e a restante informacao'),
             write(' adicional, foi arquivada:'),nl,
             write(documento(Z,d(A,M,T,L,Y,C,R))),nl.


/* Generation of new categories */

generate_cat(N,C,M):-N1 is N+1,dialogue(crie,1,N1,_),
     ?fr(p,N1,title(_,Y)),
     cat(_,_,Y,P),K is P*10,
     down(L,Y),length(L,S),W is K+S+1,
     archive_cat(Y,P,C,W),
     write('A nova categoria ficou inserida no sistema'),
     write(' de classificacao, e recebeu o numero '),
     write(W),write('.'),nl.


/* Classification system for Artificial Intelligence */


cat('computer sciences',0,
    'computer sciences applications',1).
cat('computer sciences',0,software,2).
cat('computer sciences',0,'mathematics of computation',3).
cat('computer sciences applications',1,hummanities,11).
cat('computer sciences applications',1,
    'artificial intelligence',12).
cat(software,2,'programming languages',21).
cat(software,2,processors,22).
cat('mathematics of computation',3,metatheory,31).
cat(humanities,11,
    'language translation and linguistics',111).
cat('artificial intelligence',12,
    'artificial intelligence applications',121).
```

```
cat('artificial intelligence',12,
    'artificial intelligence tools',122).
cat(metatheory,31,logic,311).
cat(metatheory,31,'formal languages',312).
cat(logic,311,'computational logic',3111).
cat('artificial intelligence applications',121,
    'mathematics,science,and engineering aids',1211).
cat('artificial intelligence applications',121,
    'automatic theorem proving',1212).
cat('artificial intelligence applications',121,
    robots,1213).
cat('artificial intelligence applications',121,
    'machine vision',1214).
cat('artificial intelligence applications',121,
    'natural language systems',1215).
cat('artificial intelligence applications',121,
    'information processing psychology',1216).
cat('artificial intelligence tools',122,
    'theory of heuristic methods',1221).
cat('artificial intelligence tools',122,
    'modelling and representation of knowledge',1222).
cat('artificial intelligence tools',122,
    common-sense reasoning,
    deduction and problem solving',1223).
cat('artificial intelligence tools',122,
    'artificial intelligence systems and languages',1224).
cat('programming languages',21,
    'artificial intelligence systems and languages',1224).
cat(processors,22,compilers,221).
cat(processors,22,interpreters,222).
cat('information processing psychology',1216,
    'induction and hypothesis formation',12161).
cat('information processing psychology',1216,
    'learning and adaptive systems',12162).
cat('modelling and representation of knowledge',1222,
    'frame systems',12221).
cat('modelling and representation of knowledge',1222,
    'predicate calculus',12222).
cat('common-sense reasoning,deduction and problemsolving',
    1223,'puzzle solving',12231).
cat('common-sense reasoning,deduction and problem
    solving',1223,'question answering',12232).
cat('common-sense reasoning,deduction and problem
    solving',1223,'common-sense reasoning',12233).
cat('common-sense reasoning,deduction and problem
    solving',1223,planning,12234).
cat('artificial intelligence systems and languages',1224,
    'pop-2',12241).
cat('artificial intelligence systems and languages',1224,
    planner,12242).
cat('artificial intelligence systems and languages',1224,
    qlisp,12243).
```

```
cat('artificial intelligence systems and languages',1224,
    conniver,12244).

cat('artificial intelligence systems and languages',1224,
    prolog,12245).
cat('common-sense reasoning',12233,'resolution principle',
    122331).
cat('common-sense reasoning',12233,
    'ad hoc methods',122332).



:-end.
```

APPENDIX 4: EXAMPLES OF DIALOGUES

1. SIMPLE QUESTIONS

¦ Qual e´ o autor da "Psychology of computer vision"?
[Who is the author of "Psychology of computer vision"?]
winston.

¦ Qual o livreiro da "Logic in Linguistics"?
[Who is the publisher of "Logic in Linguistics"?]
"cambridge press".

¦ E´ "Human problem solving" um livro ?
[Is "Human problem solving" a book?]
Sim.
[Yes.]

¦ Quem e´ o autor da "Artificial Intelligence" ?
[Who is the author of "Artificial Intelligence"?]
winston.

¦ Por quem e´ editada a "Computational Semantics"?
[Who is "Computational Semantics" published by?]
pela "north-holland".
[By "north-holland".]

¦ Onde esta´ editada a "Introduction to logic" ?
[By whom is "Introduction to logic" published?]
na "oxford press".
[By "oxford press".]

¦ Onde foi editada a "A machine oriented logic based on the resolution principle" ?
[By whom was "A machine oriented logic based on the resolution principle" published?]
no "journal of acm".
[By "journal of acm".]

¦ Quando foi publicada a "Logic for problem solving" ?
[When was "Logic for problem solving" published?]

em 1974.
[In 1974.]

¦ "Psychology of Computer Vision" tem como autor Winston.
["Psychology of Computer Vision has as publisher Winston.]
Concordo.
[I agree.]

¦ "Mcgraw hill" e´ o livreiro da "Psychology of computer vision".
["Mcgraw hill" is the publisher of "Psychology of computer vision".]
Concordo.
[I agree.]

¦ O "The case for case" e´ um artigo escrito pelo Fillmore.
["The case for case" is a paper written by Fillmore.]
Concordo.
[I agree.]

¦ O "The case for case" nao e´ um livro escrito pelo Fillmore.
["The case for case" is not a book written by Fillmore.]
Concordo.
[I agree.]

¦ O "The case for case" nao e´ o livro do Schank.
["The case for case" is not the book of Schank.]
Concordo.
[I agree.]

¦ Weizenbaum e´ o escritor do "Computer power and human reason".
[Weizenbaum is the writer of "Computer power and human reason".]
Concordo.
[I agree.]

¦ Quais os titulos da "Cambridge press"?
[What are the titles of "Cambridge press"?]
"logic in linguistics".

¦ Quais os titulos da Elsevier?
[What are the titles of Elsevier?]
"A computational model of skill and acquisition" e "computer-based medical consultations:MYCIN".

¦ Ha´ algum livro escrito por Hewitt ?
[Is there any book written by Hewitt?]
Nao.

[No.]

¦ Em que ano foi publicada a "Artificial intelligence ?
[In what year was "Artificial intelligence" published?]
em 1977.

¦ "Scripts,plans,goals and understanding" e´ um livro do "Lawrence Erlbaum".
["Scripts,plans,goals and understanding" is a book of "Laurence Erlbaum".]
Concordo.
[I agree.]

¦ Quem publicou a "Understanding Natural Language"?
[Who published "Understanding Natural Language"?]
"academic press".

¦ Que livro escreveu Nilsson ?
[What book wrote Nilsson?]
"problem solving methods in artificial intelligence".

¦ Que livro escreveu Nilsson que foi editado pela Elsevier?
[What book wrote Nilsson that was published by Elsevier?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Que livro do Nilsson foi editado pela "Mcgraw hill"?
[What book by Nilsson was published by "Mcgraw hill"?]
"problem solving methods in artificial intelligence".

¦ Que livros publicou a "Mcgraw Hill"?
[What books published "Mcgraw Hill"?]
"psychology of computer vision", "problem solving methods in artificial intelligence" e "ai:the heuristic programming approach".

¦ E´ "The case for case" um artigo ?
[Is "The case for case" a paper?]
Sim.
[Yes.]

¦ Qual o titulo do Back ?
[What is the title of Back?]
"universals in linguistic theory".

¦ Quem e´ o autor de "Logic programming and compiler writing" ?
[Who is the author of "Logic programming and compiler writing"?]
warren.

¦ A "Artificial Intelligence" e´ um livro do Schank .
[["Artificial Intelligence" is a book by Schank.]
Nao concordo.
[I don´t agree.]

¦ Quais os livros da Elsevier ?
[What are the books of Elsevier?]
"a computational model of skill and acquisition" e "computer-based medical consultations:MYCIN".

¦ "A framework for representing knowledge" e´ um artigo do Fikes .
["A framework for representing knowledge" is a paper by Fikes.]
Nao concordo.
[I don´t agree.]

¦ "Artificial intelligence and learning strategies" e´ um artigo do Brown .
["Artificial intelligence and learning strategies" is a paper by Brown.]
Concordo.
[I agree.]

¦ A "Artificial intelligence" e´ um livro do Nilsson que foi publicado em 1971 .
["Artificial intelligence" is a book by Nilsson that was published in 1971.]
Nao concordo.
[I don´t agree.]

¦ Quais sao as publicacoes?
[What are the documents?]
livros e artigos.
[books and papers.]

¦ Qual a classe da "MARGIE:memory analysis response generation and inference on english"?
[What is the type of "MARGIE:memory analysis response generation and inference on english"?]
artigos.
[papers.]

¦ Quais os titulos de Rulifson, Mcdermott e Burstall?
[What are the titles of Rulifson, Mcdermott and Burstall?]
"QA4:a procedural calculus for intuitive reasoning", "the CONNIVER reference manual" e "programming in POP-2".

¦ Quais sao os titulos publicados em 1978?
[What are the titles published in 1978?]
"automated theorem proving:a logical basis" e "artificial intelligence and learning strategies".

¦ Que artigos foram publicados em 1976?
[What were the papers published in 1976?]
"GEOM:a PROLOG geometry theorem prover", "project planning using a hierarchic non-linear planner" e "a PROLOG-like interpreter for non-horn clauses".

¦ Quantos titulos estao arquivados?
[How many titles are stored?]
46.

¦ Quantos livros estao arquivados?
[How many books are stored?]
21.

¦ Quantos livros existem?
[How many books are there?]
21.

¦ Quantos artigos existem?
[How many papers are there?]
25.

¦ Qual e´ o autor de "Artificial Intelligence" e "Computer models of thought and language"?
[Who is the author of "Artificial Intelligence" and "Computer models of thought and language"?]
A sua pergunta e´ indefinida!
Nao encontro nenhuma entidade que a satisfaca.
[Your question is undefined!
I don´t find any appropriate entity.]

¦ Qual e´ o autor de "Learning structural descriptions from examples" e "Artificial Intelligence"?
[Who is the author of "Learning structural descriptions from examples" and "Artificial Intelligence"?]
winston.

¦ E´ "Holt,Rinehart and Winston" o livreiro da "Learning and executing generalized robot plans"?
[Is "Holt,Rinehart and Winston" the publisher of "Learning and executing generalized robot plans"?]
Nao.
[No.]

¦ Quais os livros publicados apos 1976?
[What are the books published after 1976?]
"scripts,plans,goals and understanding", "artificial intelligence", "logic in linguistics", "automated theorem proving:a logical basis" e "the process of question answering".

¦ Diga-me se "Logic for Problem Solving" e´ um livro?
[Tell me if "Logic for Problem Solving" is a book?]

Nao.
[No.]

¦ Quais sao os livros cujo livreiro e´ a "North-Holland"?
[What are the books whose publisher is "North-Holland"?]
"computational semantics" e "automated theorem proving:a logical basis".

¦ Qual a data da "Exploiting the properties of function to control search "?
[What is the date of "Exploiting the properties of function to control search"?]
1977.

¦ "Logic in linguistics" e "Artificial intelligence" sao livros.
["Logic in linguistics" and "Artificial intelligence" are books.]
Concordo.
[I agree.]

¦ A "Artificial intelligence" e a "A machine oriented logic based on the resolution principle" sao artigos.
["Artificial intelligence" and "A machine oriented logic based on the resolution principle" are papers.]
Concordo.
[I agree.]

¦ Quais os livros publicados entre 1976 e 1978?
[What are the books published between 1976 and 1978?]
"scripts,plans,goals and understanding", "artificial intelligence", "logic in linguistics" e "the process of question answering".

¦ Quais os artigos publicados entre 1974 e 1976?
[What are the papers published between 1974 and 1976?]
"les grammaires de metamorphose".

¦ Nilsson nao escreveu a "Psychology of computer vision".
[Nilsson did not write "Psychology of computer vision".]
Concordo.
[I agree.]

¦ Elsevier nao escreveu a "Artificial Intelligence".
[Elsevier did not write "Artificial Intelligence".]
Concordo.
[I agree.]

¦ Ninguem escreveu a "Artificial Intelligence".
[Nobody wrote "Artificial Intelligence".]

Nao concordo.
[I don´t agree.]

¦ "Planning in a hierarchy of abstraction spaces" nao e´ um artigo.
["Planning in a hierarchy of abstraction spaces" is not a paper.]
Nao concordo.
[I don´t agree.]

¦ "Computational semantics" e "The process of question answering" sao dois livros.
["Computational semantics" and "The process of question answering" are two books.]
Concordo.
[I agree.]

¦ Qual o livro que foi escrito pelo Nilsson?
[What is the book that was written by Nilsson?]
"problem solving methods in artificial intelligence".

¦ Quais os titulos cujo autor e´ Slagle?
[What are the titles whose author is Slagle?]
"ai:the heuristic programming approach".

¦ Sera´ que Charniak e´ um livreiro?
[Is it the case that Charniak is a publisher?]
Nao.
[No.]

¦ A "Artificial intelligence" foi publicada pela "Mcgraw Hill" em 1971.
["Artificial intelligence" was published by "Mcgraw Hill" in 1971.]
Nao concordo.
[I don´t agree.]

¦ "Computational Semantics" tem a "North-Holland" como livreiro.
["Computational Semantics" has "North-Holland" as publisher.]
Concordo.
[I agree.]

¦ Quais os livros publicados desde 1977?
[What are the books published after 1977?]
"scripts,plans,goals and understanding", "artificial intelligence", "logic in linguistics", "automated theorem proving:a logical basis" e "the process of question answering".

¦ Quais sao as categorias da "Artificial intelligence"?
[What are the categories of "Artificial intelligence"?]

12--"artificial intelligence".

¦ Classifique-me "Les grammaires de metamorphose"!
[Classify to me "Les grammaires de metamorphose"!]
"les grammaires de metamorphose" e´ conhecido,e esta´ classificado nas categorias:
["les grammaires de metamorphose" is known, and it is classified under the categories:]
111--"language translation and linguistics", 1215--"natural language systems" e 312--"formal languages".

¦ Classifique a "Artificial intelligence"!
[Classify "Artificial intelligence"!]
"artificial intelligence" e´ conhecido, e esta´ classificado nas categorias:
["artificial intelligence" is known, and it is classified under the following categories:]
12--"artificial intelligence".

¦ De-me a classificacao de "Logic for problem solving"!
[Give to me the classification of "Logic for problem solving"!]
1223--"common-sense reasoning,deduction and problemsolving" e 3111--"computational logic".

¦ Quero classificar "Psychology of computer vision"!
[I want to classify "Psychology of computer vision"!]
"psychology of computer vision" e´ conhecido, e esta´ classificado nas categorias:
["psychology of computer vision" is known, and it is classified under the following categories:]
1214--"machine vision", 1216--"information processing psychology" e 1222--"modelling and representation of knowledge".

¦ Existe alguma classificacao para "The case for case"?
[Is there any classification for "The case for case"?]
Nao.
[No.]

¦ Conhece alguma classificacao para "The CONNIVER reference manual" ?
[Do you know any classification for "The CONNIVER reference manual"?]
Nao.
[No.]

¦ Crie PROLOG !
[Generate PROLOG !]
"prolog" e´ conhecida!
["prolog" is known!]

```
¦ Quero criar CONNIVER !
     [I want to generate CONNIVER !]
"conniver" e´ conhecida!
     ["conniver" is known !]

¦ Qual e´ a classificacao de "Artificial intelligence"?
     [What is the classification of "Artificial
intelligence"?]
12--"artificial intelligence".

¦ Como classifica a "Computational semantics" ?
     [How do you classify "Computational semantics"?]
no 1215--"natural language systems" e 312--"formal
languages".

¦ Qual é´ a bibliografia de "Computational semantics" ?
     [What is the bibliography of "Computational
semantics"?]
Os cinco primeiros sao:
     [The five first ones are:]
livro no. 3--"understanding natural language",
livro no. 6--"a computational model of skill and
acquisition",
livro no. 9--"toward a model of children´s story
comprehension",
artigo no. 26--"semantic networks:their computation and
use for understanding english sentences" e
artigo no. 28--"MARGIE:memory analysis response
generation and inference on english".
Existem mais 2!
Quer mais?
     [There are more two!
      Do you want more?]
Nao.
     [No.]
Esta´ bem!
     [All right!]

¦ Qual e´ a bibliografia para a "Artificial intelligence"
?
     [What is the bibliography for "Artificial
intelligence"?]
Os cincc primeiros sao:
     [The five first ones are:]
livro no. 2--"problem solving methods in artificial
intelligence",
livro no. 6--"a computational model of skill and
acquisition",
livro no. 7--"human problem solving",
livro no. 10--"computer power and human reason" e
artigo no. 23--"STRIPS:a new approach to the application
of theorem proving to problem solving".
Existem mais 3!
Quer mais?
```

[There are more 3!
Do you want more?]
Nao.
[No.]
Esta´ bem!
[All right!]

¦ Quais os livros classificados em 1224 ?
[What are the books classified under 1224?]
A sua pergunta e´ indefinida!
Nao encontro nenhuma entidade que a satisfaca.
[Your question is undefined!
I don´t find any appropriate entity!]

¦ Quais os artigos classificados em 1214 ?
[What are the papers classified under 1214?]
"a framework for representing knowledge".

¦ Quais os artigos classificados em prolog ?
[What are the papers classified under prolog?]
"a PROLOG-like interpreter for non-horn clauses" e "implementing PROLOG-compiling logic programs".

¦ Quais os livros relacionados com "Computational Semantics" ?
[What are the books related with "Computational Semantics"?]
"understanding natural language", "a computational model of skill and acquisition" e "toward a model of children´s story comprehension".

¦ Quais os artigos referenciados em "Logic for problem solving" ?
[What are the papers referred in "Logic for problem solving"?]
"description and theorectical analysis (using schemata) of planner", "STRIPS:a new approach to the application of theorem proving to problem solving", "QA4"a procedural calculus for intuitive reasoning", "a preliminary QLISP manual" e "a machine oriented logic based on the resolution principle".

¦ Quais os artigos referenciados com "Logic for problem solving" ?
[What are the papers referred with "Logic for problem solving"?]
"GEOM:a PROLOG geometry theorem prover", "a PROLOG-like interpreter for non-horn clauses", "implementing PROLOG-compiling logic programs" e "logic programming and compiler writing".

¦ Quais os artigos sobre prolog ?
[What are the papers about prolog?]
"a PROLOG-like interpreter for non-horn clauses" e

```
"implementing PROLOG-compiling logic programs".

¦ Qual o nome de 1224 ?
     [What is the name of 1224?]
"artificial intelligence systems and languages".

¦ O que significa 1214 ?
     [What means 1214?]
"machine vision".

¦ O que diz 1222 ?
     [What says 1222?]
"modelling and representation of knowledge".

¦ O que quer dizer 111 ?
     [What means 111?]
"language translation and linguistics".

¦ O que atribui `a 1223 ?
     [What do you assign to 1223?]
"common-sense reasoning,deduction and problem solving".

¦ Que nome atribui `a 12222 ?
     [What name do you assign to 12222?]
"predicate calculus".

¦ O que chama `a 111 ?
     [What do you call to 111?]
"language translation and linguistics".

¦ Existe alguma categoria para prolog ?
     [Is there any category for prolog?]
Sim.
     [Yes.]

¦ E´ prolog uma categoria ?
     [Is prolog a category?]
Sim.
     [Yes.]

¦ E´ 1214 uma categoria ?
     [Is 1214 a category?]
Sim.
     [Yes.]

¦ 1214 e´ uma categoria ?
     [Is 1214 a category?]
Sim.
     [Yes.]

¦ Estao "Artificial intelligence systems and languages" e
prolog ligadas ?
     [Are "Artificial intelligence systems and languages"
and prolog linked?]
```

```
Nao compreendo esta frase.
     [I don´t understand this sentence.]

¦ Estao ligadas "Artificial intelligence systems and
languages" e prolog ?
     [Are "Artificial intelligence systems and languages"
and prolog linked?]
Nao compreendo esta frase.
     [I don´t understand this sentence.]

¦ Esta´ logic ligada com Prolog ?
     [Is logic linked with Prolog?]
Nao compreendo esta frase.
     [I don´t understand this sentence.]

¦ Logic esta´ ligada com Prolog ?
     [Is Logic linked with Prolog?]
Nao.
     [No.]

¦ A Logic esta´ ligada `a "Computational logic" .
     [Logic is linked to "Computational logic".]
Concordo.
     [I agree.]

¦ Quais as categorias dependentes de "Artificial
intelligence systems and languages" ?
     [What are the categories under "Artificial
intelligence systems and languages"?]
pop-2, planner, qlisp, conniver e prolog.

¦ Quais as categorias abaixo de "Common-sense reasoning"
?
     [What are the categories below "Common-sense
reasoning"?]
"resolution principle" e "ad hoc methods".

¦ Quais as categorias dominantes de Processors ?
     [What are the categories above Processors?]
"computer sciences" e "software".

¦ Quais as categorias acima de Prolog ?
     [What are the categories above Prolog?]
Os cinco primeiros sao:
     [The first five ones are:]
"computer sciences", "computer sciences applications",
software, artificial intelligence" e "programming
languages".
Existem mais 2!
Quer mais?
     [There are more two!
      Do you want more?]
Sim.
     [Yes.]
```

Os restantes sao:"artificial intelligence tools" e "artificial intelligence systems and languages".

¦ Quais as categorias que dominam Processors ?
[Which are the categories that dominate Processors?]
"computer sciences" e software.

¦ Quais as categorias ate´ "frame systems" ?
[What are the categories till "frame systems"?]
"modelling and representation of knowledge", "computer sciences", "computer sciences applications", "artificial intelligence" e "artificial intelligence tools".

¦ Qual a categoria que domina "computational logic" ?
[Which is the category that dominates "computational logic"?]
"computer sciences".

¦ Qual a categoria que esta´ acima de Metatheory ?
[Which is the category that is above Metatheory?]
"computer sciences".

¦ Qual a categoria que esta´ junto `a "computational logic"?
[Which is the category that is next to "computational logic"?]
logic.

¦ Informe-me sobre "Psychology of computer vision" !
[Inform to me about "Psychology of computer vision"!]
"psychology of computer vision" e´ conhecido, e possui a seguinte informacao:
["psychology of computer vision" is known,
has appended the following information:]
livro no.1,
autor: winston,
livreiro: mcgraw hill,
ano de publicacao: 1975,
classificacao:
1214--"machine vision", 1216--"information processing psychology" e 1222--"modelling and representation of knowledge"
referencias:
livro no. 3--"understanding natural language" e
livro no. 4--"computer models of thought and language"..

¦ Qual a informacao sobre "Computational semantics" ?
[What is the information about "Computational semantics"?]
livro no.45,
autor: charniak,
livreiro: north-holland,
ano de publicacao: 1976, classificacao:

1215--"natural language systems" e 312--"formal languages".,
referencias:
livro no. 3--"understanding natural language",
livro no. 6--"a computational model of skill and acquisition",
livro no. 9--"toward a model of children´s story comprehension",
artigo no. 26--"semantic networks: their computation and use for understanding english sentences",
artigo no. 28--"MARGIE:memory analysis response generation and inference on english",
artigo no. 31--"the case for case" e
artigo no. 43--"PLANNER:a language for proving theorems in robots"..

¦ Qual e´ a informacao sobre "PLANNER:a language for proving theorems in robots" ?
[What is the information about "PLANNER:a language for proving theorems in robots"?]
artigo no.43,
autor: hewitt,
livreiro: ijcai,
ano de publicacao: 1969,
classificacao:
1212--"automatic theorem proving", 1213--"robots" e 12242--"planner".

¦ Qual o titulo de Bach ou Schank ?
[What is the title of Bach or Schank?]
"universals in linguistic theory".

¦ Qualquer autor publica na Freeman .
[Any author publishes at Freeman.]
Nao concordo.
[I don´t agree.]

¦ Nem todos os autores publicam na "Academic press" .
[Not all authors publish at "Academic press".]
Concordo.
[I agree.]

¦ Todos os autores sao editores .
[All the authors are editors.]
Concordo.
[I agree.]

¦ Todos os autores sao livreiros .
[All the authors are publishers.]
Nao concordo.
[I don´t agree.]

¦ Qual o titulo cujo autor e´ Slagle ?
[What is the title whose author is Slagle?]

A sua pergunta e´ indefinida!
Nao encontro nenhuma entidade que a satisfaca.
[Your question is undefined!
I don´t find any appropriate entity.]

¦ Kowalski e´ o autor cujo artigo foi publicado em 1974 .
[Kowalski is the author whose paper was published in 1974.]
A sua frase pressupoe outros factos, logo um contexto.
[Your sentence presupposes other facts, therefore a context.]
Como nao possuo informacao sobre o que foi dito anteriormente, a sua frase e´ ambigua.
[As I have no information about what was said previously, your sentence is undefined.]
E, portanto, nao consigo responder-lhe!
[And, therefore, I cannot answer you!]

¦ Kowalski e´ um autor o qual escreveu "Logic for problem solving" .
[Kowalski is an author who wrote "Logic for problem solving".]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Qual o livreiro onde foi editado "Artificial Intelligence" ?
[Who is the publisher where "Artificial Intelligence" was published?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Que livros nao estao publicados na "Mcgraw Hill" , Elsevier , "North-Holland" e "Academic Press" ?
[What books are not published at "Mcgraw Hill", Elsevier, "North-Holland" and "Academic Press"?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Que livro tem 1974 ?
[What book has 1974?]
Nenhuma entidade satisfaz a sua pergunta.
[No entity satisfies your question.]

¦ Cada livro tem um autor .
[Each book has an author.]
Nao concordo.
[I don´t agree.]

¦ Cada pessoa tem um titulo .
[Each person has a title.]
Concordo.
[I agree.]

¦ Bach nao tem nenhum titulo .
[Bach has no title.]
Nao concordo.
[I don´t agree.]

¦ Todos os livros tem um autor .
[All the books have an author.]
Nao concordo.
[I don´t agree.]

¦ Todos os livros tem autores .
[All the books have authors.]
Nao concordo.
[I don´t agree.]

¦ A data que "Logic for problem solving" tem e´ 1974 .
[The date that "Logic for problem solving" has is 1974.]
Nao concordo.
[I don´t agree.]

¦ Quem publicou "Psychology of computer vision" , "Understanding Natural Language" e "Programming in POP-2"?
[Who published "Psychology of computer vision", "understanding Natural Language" and "Programming in POP-2"?]
"mcgraw hill".

¦ Quem publicaram "A preliminary QLISP manual" , "A framework for representing knowledge" e "Human problem solving" ?
[Who publish "A preliminary QLISP manual", "A framework for representing knowledge" and "Human problem solving"?]
sri, "mit ai lab" e "prentice hall".

¦ Quem e´ o autor de "Artificial intelligence" ou "Computational semantics" ?
[Who is the author of "Artificial intelligence" or "Computational semantics"?]
winston.

¦ Lehnert escreveu "The process of question answering" , e Charniak escreveu "Computational semantics" .
[Lehnert wrote "The process of question answering", and Charniak wrote "Computational semantics".]
Para a primeira proposicao: Concordo.
[For the first sentence: I agree.]
E, para a segunda proposicao: Concordo.
[And, for the second sentence: I agree.]

¦ Que informacao tem sobre "Logic for problem solving" ?
[What information have you about "Logic for problem

solving"?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ "Logic for problem solving" tem um autor .
["Logic for problem solving" has an author.]
Nao concordo.
[I don´t agree.]

¦ Alguns autores publicam na Elsevier .
[Some authors are published by Elsevier.]
Concordo.
[I agree.]

¦ Alguem publica na "Edinburgh Press" .
[Somebody is published by "Edinburgh Press".]
Concordo.
[I agree.]

¦ Conhece algum titulo para Slagle ?
[Do you know any title for Slagle?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Todos os autores publicados tem um livro .
[All the published authors have a book.]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Foi o livro escrito pelo Back ?
[Was the book written by Back?]
Nao compreendo esta frase.
[I don´t understand this sentence.]

¦ Foi escrito o livro pelo Fikes ?
[Was written the book by Fikes?]
Nao.
[No.]

¦ O livro foi escrito pelo Winston .
[The book was written by Winston.]
Concordo.
[I agree.]

¦ A "Academic press" e´ a livreira do artigo .
["Academic press" is the publisher of the paper.]
A sua frase pressupoe outros factos, logo um contexto.
[Your sentence presupposes other facts, therefore a context.]
Como nao possuo informacao sobre o que foi dito anteriormente, a sua frase e´ ambigua.
[As I have no information about what was said previously, your sentence is undefined.]
E, portanto, nao consigo responder-lhe!

[And, therefore, I cannot answer you!]

¦ O livreiro e´ a "Macgraw Hill" .
[The publisher is "Mcgraw Hill".
Nao compreendo esta frase.
[I don´t understand your sentence.]

¦ Colmerauer nao escreveu um livro .
[Colmerauer did not write a book.]
Concordo.
[I agree.]

¦ Brown e´ o autor do artigo .
[Brown is the author of the paper.]
A sua frase pressupoe outros factos, logo um contexto.
[Your sentence presupposes other facts, therefore a context.]
Como nao possuo informacao sobre o que foi dito anteriormente, a sua frase e´ ambigua.
[As I have no information about what was said previously, your sentence is undefined.]
E, portanto, nao consigo responder-lhe!
[And, therefore, I cannot answer you!]

¦ "Computational semantics" e´ o livro .
["Computational semantics" is the book.]
Concordo.
[I agree.]

¦ Quem e´ o autor do artigo ?
[Who is the author of the paper?]
A sua pergunta e´ indefinida!
[Your question is undefined!]
Nao encontro nenhuma entidade que a satisfaca.
[I don´t find any appropriate entity.]

¦ O livro foi publicado em 1978 .
[The book was published in 1978.]
Concordo.
[I agree.]

¦ Sacerdoti e´ o tipo que escreveu "Logic for problem solving" .
[Sacerdoti is the man who wrote "Logic for problem solving".]
Nao concordo.
[I don´t agree.]

¦ A pessoa que escreveu "Logic for problem solving" e´ o Kowalski .
[The person who wrote "Logic for problem solving" is Kowalski.]
Concordo.
[I agree.]

¦ Sacerdoti e´ um autor ?
[Sacerdoti is an author?]
Sim.
[Yes.]

2. DIALOGUES

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please, write facts, commands or questions.]
¦ Michie e´ um autor?
[Is Michie an author?]

Nao compreendo esta frase porque a palavra michie e´ desconhecida!
[I don´t understand this sentence because the word michie is unknown!]
Por favor, preste atencao `a frase que escreveu!
[Please, observe with care the sentence you have typed!]
Existe algum erro sintatico na escrita da sua frase?
[Is there any syntactical mistake?]

¦ Nao existe.
[There is none.]

Esta´ bem!
[All right!]
A palavra desconhecida e´ um nome proprio?
[Is the unknown word michie a proper noun?]

¦ E´.
[It is.]

Qual e´ o genero de michie ?
[What is the gender of michie?]

¦ Masculino.
[Masculine.]

A qual dos tipos, autor, titulo, livreiro, ou categoria, pertence a palavra michie ?
[To which type, author, title, publisher or category, does the word michie belong?]

¦ autor.
[author.]

A resposta `a sua pergunta sobre michie e´:
[The answer to your question about michie is:]

Nao.
[No.]
E que mais?
[And what else?]

¦ Adeus.
[Goodbye.]

Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please, write facts, commands or questions.]

¦ Qual e´ a bibliografia para "Artificial Intelligence"?
[What is the bibliography of "Artificial Intelligence"?]

Os cinco primeiros sao:
[The first five are:]
livro no. 2--"problem solving methods in artificial intelligence",
livro no. 6--"a computational model of skill and acquisition",
livro no. 7--"human problem solving",
livro no. 10--"computer power and human reason" e
artigo no. 23--"STRIPS:a new approach to the application of theorem proving to problem solving".
Existem mais 3 documentos!
[There are three more documents!]
Quer mais documentos?
[Do you want any more documents?]

¦ Quero mais dois documentos.
[Yes.I want two more documents.]

Os restantes sao:
[The remainder are:]

artigo no. 25--"the CONNIVER reference manual" e
artigo no. 27--"planning in a hierarchy of abstraction spaces".
E que mais?
[And what else?]

¦ Adeus.
[Goodbye.]

Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please, write facts, commands or questions.]

¦ Arquive "Experimental learning model"!
[Archive "Experimental learning model"!]

"experimental learning model" e´ desconhecido!
["experimental learning model" is unknown!]
Gostaria de o arquivar!
[I would like to archive it!]
Por favor, conhece a classificacao de "experimental learning model" ?
[Please do you know the classification for "experimental learning model"?]

¦ Sim, conheco.
[Yes.I know.]

Por favor, quais sao as categorias do documento?
[Please what are the categories of the document?]

¦ 111 e 1214.

Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ "Psychology of Computer Vision".

"psychology of computer vision" e´ conhecido,e esta´ classificado nas categorias:

["psychology of computer vision" is known, and it is classified under the categories:]
1214--"machine vision", 1216--"information processing psychology" e 1222--"modelling and representation of knowledge".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ Nao tenho mais nenhuma.
[I have no more.]

Esta´ bem!
[All right.]
Por favor, qual o nome do autor do documento?
[Please what is the name of the author of the document?]

¦ brazdil,dai e 1978.

Por favor, qual e´ o tipo do documento?
[Please what is the type of the document?]

¦ Artigo.
[Paper.]

O novo documento no.47, e a restante informacao adicional, foi arquivada:
[The new document no.47, and the additional information has been stored!]
documento(artigo,d(brazdil,47,experimental learning model,dai,1978,[111,1214],[1]))
E que mais?
[And what else?]

¦ Apague "Experimental learning model"!
[Delete "Experimental learning model"!]

"experimental learning model" e´ conhecido, e foi apagado!
["experimental learning model" is known and was deleted!]
E que mais?
[And what else?]

¦ Adeus.
[Goodbye.]

Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

```
¦ :-ola.

Ola´.
     [Hello.]
Vamos iniciar a conversa!
     [Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
     [Please write facts, commands or questions.]

¦ Classifique "Explorations in cognition"!
     [Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
     ["explorations in cognition is unknown!]
Quer classifica´-lo?
     [Do you want to classify it?]

¦ Sim.
     [Yes.]

Vamos classifica´-lo!
     [Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento
em questao.
     [Please give me the title of one reference from the
document in question.]

¦ Desisto.
     [I give up.]

Ok.Vamos desviar-nos desta conversa!
     [Ok. Let us shift from this dialogue!]
E que mais?
     [And what else?]

¦ Adeus.
     [Goodbye.]

Ok,esta conversa terminou!
     [Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
     [Goodbye, and see you soon!]


¦ :-ola.

Ola´.
     [Hello.]
Vamos iniciar a conversa!
     [Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
     [Please write facts, commands or questions.]
```

¦ Classifique "Explorations in cognition"!
[Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
["explorations in cognition" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim.
[Yes.]

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ "Artificial intelligence".

"artificial intelligence" e´ conhecido,e esta´ classificado nas categorias:
["artificial intelligence" is known, and is classified under the categories:]
12--"artificial intelligence".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ Desisto.
[I give up.]

Ok.Vamos desviar-nos desta conversa!
[Ok. Let us shift from this dialogue!]

E que mais?
[And what else?]

¦ Classifique "Explorations in cognition"!
[Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
["explorations in cognition" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim.
[Yes.]

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de outra referencia do documento em questao.

[Please give me the title of one reference from the document in question.]

¦ Nao tenho nenhuma.
[I have none.]

Esta´ bem!
[All right!]
Ja´ que nao consegui obter nenhuma referencia, nao sei dar-lhe qualquer sugestao!
[As I could not get any reference, I don´t know how to give any suggestion!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Nao.
[No.]

Esta´ bem!
[All right!]
E que mais?
[And what else?]

¦ Adeus.
[Goodbye.]

Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please write facts, commands or questions.]

¦ Classifique "Explorations in cognition"!
[Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
["explorations in cognition" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Esta´ bem.
[It is all right.]

Vamos classifica´-lo!

[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ Nao tenho nenhuma.
[I have none.]

Esta´ bem!
[All right!]
Ja´ que nao consegui obter nenhuma referencia, nao sei dar-lhe qualquer sugestao!
[As I could not get any reference, I don´t know how to give you any suggestion!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Nao quero.
[I don´t want.]

Esta´ bem!
[All right!]
E que mais?
[And what else?]

¦ Classifique "Computational Semantics"!
[Classify "Computational Semantics"!]

"computational semantics" e´ conhecido,e esta´ classificado nas categorias:
["computational semantics" is known, and is classified under the categories:]
1215--"natural language systems" e 312--"formal languages".
E que mais?
[And what else?]

¦ Classifique "Explorations in cognition"!
[Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
["explorations in cognition" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim, quero.
[Yes, I want.]

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the

document in question.]

¦ "Computational semantics".

"computational semantics" e´ conhecido,e esta´ classificado nas categorias:
["computational semantics" is known, and is classified under the categories:]
1215--"natural language systems" e 312--"formal languages".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ "Computer models of thought and language".

"computer models of thought and language" e´ conhecido,e esta´ classificado nas categorias:
["computer models of thought and language" is known, and is classified under the categories:]
1215--"natural language systems" e 1222--"modelling and representation of knowledge".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ Nao tenho mais.
[I have no more.]

Esta´ bem!
[All right!]
Vamos utilizar o metodo de classificacao que consiste em sugerir
como categorias para o documento em questao,
as obtidas das 2 referencias fornecidas pelo utilizador!
[Let us adopt as classification method the one that consists of suggesting as categories of the document in question, the ones obtained from two references given by the user!]
Assim,proponho que o documento fique classificado nas categorias seguintes: 312, 1215 e 1222.
[I therefore propose that the document be classified under the following categories:]
Por favor,faca a sua escolha tendo em atencao as categorias sugeridas e as que julgar mais apropriadas.
[Please choose according the suggested categories and the ones you think most appropriate.]
De-me 3 categorias no maximo!
[Give me a maximum of three!]

¦ 312 e 1215.

O documento ficou classificado nas categorias:
[The document has been classified under the categories:]
312--"formal languages" e 1215--"natural language systems".
Por favor, deseja arquivar este documento na Base de Dados?
[Please do you want to store this document in the Data Base?]

¦ Nao.
[No.]

Esta´ bem!
[All right!]
E que mais?
[And what else?]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please write facts, commands or questions.]

¦ Classifique "Experimental learning model"!
[Classify "Experimental learning model"!]

"experimental learning model" e´ desconhecido!
["experimental learning model" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Ok.

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ "Artificial intelligence".

"artificial intelligence" e´ conhecido,e esta´ classificado nas categorias:
["artificial intelligence" is known, and is classified under the categories:]
12--"artificial intelligence".
Por favor,de-me o titulo de outra referencia do documento

em questao.
[Please give me the title of another reference from the document in question.]

¦ Nao tenho mais nenhum.
[I have no more.]

Esta´ bem!
[All right!]
Vamos utilizar o metodo de classificacao que consiste em sugerir como categorias para o documento em questao, as obtidas das 1 referencias fornecidas pelo utilizador!
[Let us adopt as classification method the one that consists of suggesting as categories of the document in question, the ones obtained from one references given by the user!]
Assim,proponho que o documento fique classificado nas categorias seguintes: 12.
[I therefore propose that the document be classified under the following categories:]
Por favor,faca a sua escolha tendo em atencao as categorias sugeridas e as que julgar mais apropriadas.
[Please choose according the suggested categories and the ones that you think most appropriate.]
De-me 3 categorias no maximo!
[Give me a maximum of three!]

¦ 12,111 e 1224.

O documento ficou classificado nas categorias:
[The document has been classified under the categories:]
12--"artificial intelligence", 111--"language translation and linguistics" e 1224--"artificial intelligence systems and languages".
Por favor, deseja arquivar este documento na Base de Dados?
[Please do you want to store this document in the Data Base?]

¦ Sim.
[Yes.]

Por favor, qual o nome do autor do documento?
[Please what is the name of the author of the document?]

¦ Bradzil.

Por favor, qual o nome do livreiro do documento?
[Please what is the name of the publisher of the document?]

¦ DAI.

Por favor, qual e´ a data de publicacao do documento?
[Please what is the date of publication of the document?]

¦ 1978.

Por favor, qual e´ o tipo do documento?
[Please what is the type of the document?]

¦ Artigo.
[Paper.]

O novo documento no.47, e a restante informacao adicional, foi arquivada!
[The new document no.47, and the additional information, has been stored!]
E que mais?
[And what else?]

¦ Adeus.
[Goodbye.]

Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please write facts, commands or questions.]

¦ Classifique "Explorations in cogntion"!
[Classify "Explorations in cogntion"!]

Nao compreendo esta frase.
[I don´t understand this sentence.]
Existem 2 casos possiveis para a incompreensao da frase:
[There are two possible cases for the incomprehensible sentence:]
1)--erro(s) sintactico(s) na escrita da frase;
[syntactical mistake in writing the sentence;]
2)--construcao sintactica desconhecida do programa.
[unknown syntactical construction.]
Preste atencao ´a frase que escreveu, corrija a sua sintaxe, e repita a pergunta, por favor.
[Take care to the sentence you wrote, correct its syntax, and repete the question, please.]

E que mais?
[And what else?]

¦ Classifique "Explorations in cognition"!
[Classify "Explorations in cognition"!]

"explorations in cognition" e´ desconhecido!
["explorations in cognition" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim.
[Yes.]

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ Nao tenho nenhuma.
[I have none.]

Esta´ bem!
[All right!]
Ja´ que nao consegui obter nenhuma referencia, nao sei dar-lhe qualquer sugestao!
[As I could not get any reference, I don´t known how to give any suggestion!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim.
[Yes.]

De-me 3 categorias no maximo!
[Give me a maximum of three!]

¦ 111,1214 e 156.

A categoria 156 nao existe, e portanto e´ rejeitada!
[Category 156 does not exist, and therefore it is rejected!]
O documento ficou classificado nas categorias:
[The document has been classified under the categories:]
111--"language translation and linguistics" e
1214--"machine vision".
Por favor, deseja arquivar este documento na Base de Dados?
[Please do you want to store this document in the Data Base?]

¦ Nao.
[No.]

Esta´ bem!
[All right!]
E que mais?
[And what else?]

¦ :-ola.

Ola´.
[Hello.]
Vamos iniciar a conversa!
[Let us start the dialogue!]
Por favor, escreva factos, ordens ou perguntas.
[Please write facts, commands or questions.]

¦ Classifique "A structure for plans and behaviour"!
[Classify "A structure for plans and behaviour"!]

"a structure for plans and behaviour" e´ desconhecido!
["a structure for plans and behaviour" is unknown!]
Quer classifica´-lo?
[Do you want to classify it?]

¦ Sim, quero.
[Yes, I want.]

Vamos classifica´-lo!
[Let us classify it!]
Por favor,de-me o titulo de uma referencia do documento em questao.
[Please give me the title of one reference from the document in question.]

¦ "Computer power and human reason".

"computer power and human reason" e´ conhecido,e esta´ classificado nas categorias:
["computer power and human reason" is known, and is classified under the categories:]
1215--"natural language systems".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ "Understanding natural language".

"understanding natural language" e´ conhecido,e esta´ classificado nas categorias:
["understanding natural language" is known, and is

classified under the categories:]
111--"language translation and linguistics",
1215--"natural language systems" e 312--"formal languages".
Por favor,de-me o titulo de outra referencia do documento em questao.
[Please give me the title of another reference from the document in question.]

¦ "Computational semantics".

"computational semantics" e´ conhecido,e esta´ classificado nas categorias:
["computational semantics" is known, and is classified under the categories:]
1215--"natural language systems" e 312--"formal languages".
Vamos utilizar o metodo de classificacao que consiste em sugerir como categorias para o documento em questao, as obtidas das 3 referencias fornecidas pelo utilizador!
[Let us adopt as classification method the one that consists of suggesting as categories of the document in question, the ones obtained from three references given by the user!]
Assim,proponho que o documento fique classificado nas categorias seguintes: 111, 1215 e 312.
[I therefore propose that the document be classified under the following categories:]
Por favor,faca a sua escolha tendo em atencao as categorias sugeridas e as que julgar mais apropriadas.
[Please choose according the suggested categories and the ones you think most appropriate.]
De-me 3 categorias no maximo!
[Give me a maximum of three!]

¦ 111,1215 e 1214.

O documento ficou classificado nas categorias:
[The document has been classified under the categories:]
111--"language translation and linguistics",
1215--"natural language systems" e 1214--"machine vision".
Por favor, deseja arquivar este documento na Base de Dados?
[Please do you want to store this document in the Data Base?]

¦ Ok.

Por favor, qual o nome do autor do documento?
[Please what is the name of the author of the document?]

¦ Sacerdoti.

Por favor, qual o nome do livreiro do documento?
[Please what is the name of the publisher of the document?]

¦ Elsevier.

Por favor, qual e´ a data de publicacao do documento?
[Please what is the date of publication of the document?]

¦ 1977.

Por favor, qual e´ o tipo do documento?
[Please what is the type of the document?]

¦ Livro.
[Book.]

O novo documento no.47, e a restante informacao adicional, foi arquivada!
[The new document no.47, and the additional information, has been stored!]
E que mais?
[And what else?]

¦ Crie uma categoria.
[Create one category.]

Concordo.
[I agree.]
E que mais?
[And what else?]

¦ Crie uma categoria!
[Create one category!]

Por favor, qual o nome da nova categoria?
[Please what is the name of the new category?]

¦ "Pattern recognition".

"pattern recognition" e´ desconhecida!
["pattern recognition" is unknown!]
Por favor,debaixo de que categoria a pretende inserir?
[Please under what category do you want to insert it?]

¦ "Machine vision".

"machine vision" e´ conhecida!
["machine vision" is known!]
A nova categoria ficou inserida no sistema de

classificacao, e recebeu o numero 12141.
[The new category has been inserted in the classification system and received 12141 as a number.]
E que mais?
[And what else?]

¦ Crie "pattern recognition".
[Create "pattern recognition".]

Concordo.
[I agree.]
E que mais?
[And what else?]

¦ Crie "Pattern recognition"!
[Create "Pattern recognition"!]

"pattern recognition" e´ conhecida!
["pattern recognition" is known!]
E que mais?
[And what else?]

¦ Crie "Speech understanding systems"!
[Create "Speech understanding systems"!]

"speech understanding systems" e´ desconhecida!
["speech understanding systems" is unknown!]
Por favor,debaixo de que categoria a pretende inserir?
[Please under what category do you want to insert it?]

¦ "Natural language systems".

"natural language systems" e´ conhecida!
["natural language systems" is known!]
A nova categoria ficou inserida no sistema de classificacao, e recebeu o numero 12151.
[The new category has been inserted in the classification system and received 12151 as a number.]
E que mais?
[And what else?]
¦ Adeus.
[Goodbye.]
Ok,esta conversa terminou!
[Ok, this dialogue ended!]
Adeus, e ate´ ´a vista!
[Goodbye, and see you soon!]

APPENDIX 5: GLOSSARY OF ABBREVIATIONS USED IN PROGRAM IDENTIFIERS

A = variable for agreement

ad = adjective

adj = category of adjectives

adj_g = group of adjectives

adjs = adjectives

adver = category of adverbs

adverbs = adverbs

adv_g = group of adverbs

adv_m = mode adverb

adv_m_i = mode and interrogative adverb

adv_q = quantity adverb

affir = affirmation

arg = category of subjects (arguments)

arg_form = argument form

arq = archiving scenario

art = domain of papers

art = category of articles

aut = domain of authors

aut_for_aut = author published by

aut_of_book = author of a book

aut_of_paper = author of a paper

B = variable for the truth value

boo = domain of books

book_cat = categories of a book

book_ref = references of a book

card = cardinality

cat = domain of categories

cat = category

cat_lab = label of a category

class = class of a document

class = classification

comm = commencement

compls = category of complements

cond_output = conditional output

conj = conjunction of sentences

conj_c = coordenative conjunction

coord = coordination

crie = creating scenario

D = variable for documents and domains

def_art = category of definite articles

def_mark = definite mark

dest = destroying scenario

disj = disjunction

dom = domain

elem = element

exp = expressions

fem = feminine

find_all = find all individuals

find_all_non = find all individuals belonging to a domain, and verifies for each one whether the logical structure is true

G = variable for gender

gen = gender

info = information scenario

info = informations about a document

impk = impress data

ind_art = category of indefinite articles

int_art = category of interrogative articles

int_pron = category of an interrogative pronoun

int_prons = category of interrogative pronouns

int_rel = category of interrogative/relative pronouns

interrog = interrogation

interrog_art = interrogative article

inv_mark = inversion mark

K = variable for case

L = variable for list

lex = lexicon call

lexi = lexicon call

mas = masculine

mov_arg = moving argument

N = variable for number

nb = number

neg = negation

nega = negation call

no = noun call

nom = noun call

nos = call of nouns

n_phrase = category of noun phrases

noun = category of nouns

paper_cat = categories of a paper

paper_ref = references of a paper

per_pron = category of personnal pronouns

plu = plural

pn = category of proper nouns

pns = proper nouns

pr = property

pre_loc = pre-locution

prep = category of prepositions

prin = principal sentence

pron = category of pronouns

pron_r_i = category of relative/interrogative pronouns

pronq_r_i = category of relative/interrogative pronouns

prop = main proposition

pub = domain of publishers

pub_of_aut = an author is published by

pub_of_book = a book is published by

pub_of_tit = a title is published by

ref = domain of references

r_i_pron = relative/interrogative pronoun

rel = relative

rel_pron = relative pronoun

said = output scenario

sent = sentence

set_equal = set equality

sin = singular

s_nucleus = nucleus of a sentence

sub = subject

tit = domain of titles

tit_of_book = title of a book

tit_of_paper = title of a paper

todo_art = group of articles 'all the'

ve = verb call

verb = category of verb phrases

yea = domain of dates of publication

year_aut = date of publication of a document for an author

year_of = date of publication of a document

year_tit_pub = date of publication of a title for a publisher

BIBLIOGRAPHY

BATTANI,G.;MELONI,H.(1973)
Interpreteur du language de programmation Prolog
Univ. d'Aix-Marseille, G.I.A.

BATTANI,G.;MELONI,H.(1975)
Mise en oeuvre des contraintes phonologiques syntaxiques et semantiques dans un systeme de comprehension automatique de la parole
Univ. d'Aix-Marseille, G.I.A.

BENNETT,J. et al.(1978)
SACON: A knowledge-based consultant for structural analysis
Stanford Univ., Memo HPP-78-23

BERRY-ROGGHE,G.L.(1976)
The design of PLIDIS, a problem solving information system with German as query language
in "Advances in Natural Language Processing",
Univ. of Antwerp

BLACK,F.(1964)
A deductive question-answering system
Harward University, Ph.D. Thesis

BOBROW,D.G.(1964)
Natural language input for a computer problem solving system
Proc. of the Fall Joint Computer Conference 25

BOBROW,D.G. et al.(1976)
GUS, a frame-driven dialog system
Xerox Palo Alto Research Centre

BOBROW,D.G.;WINOGRAD,T.(1976)
An overview of KRL, a knowledge representation language
Xerox Palo Alto Research Centre

BRIABIN,V.M.;POSPELOV,D.A.(1975)
DILOS--Dialog system for information retrieval, computation and logic reference
Workshop on Dialog Systems, IIASA, Viena

BROWN,J.S. et al.(1974)
SOPHIE: a sophisticated instructional environment for teaching electronic trouble-shooting
BBN Report no.2790

BRUCE,B.(1972)
A model for temporal references and its application in a question-answering program
Artificial Intelligence Vol.3, no.1

BRUYNOOGHE,M.(1976)
An interpreter for predicate logic programs
Katholiche Universiteit Leeiven, Applied Mathematics and Programming Division, Report CN10

BUCHANAN,B.G. et al.(1976)
Applications of Artificial Intelligence for chemical inference XXII. Automatic rule formation in mass spectrometry by means of the META-DENDRAL program
Journal of the ACS

BUNDY,A. et al.(1976)
MECHO: year one
Proceed. of the AISB Conference

BURTON,R.R.(1976)
Semantic grammar: an engineering technique for construction of natural language understanding systems
BBN Report no.3453

CARBONELL,J.R.(1970)
Mixed-iniative man-computer dialogues
BBN Report no.1971

CHARNIAK,E.(1973)
Jack and Janet in search of a theory of knowledge
Proceed. of the 3IJCAI

CIPOLARO,J.K.;FINDLER,N.V.(1977)
MARSHA, the daughter of ELIZA a simple program for information retrieval in natural language
State Univ. of New York at Buffalo, Technical report no.127

CLARK,H.H.(1970)
How we understand negation
COBRE workshop on Cognitive Organization and Psychological Process, Huntingdon Beach, California

COELHO,H.;PEREIRA,L.M.(1976a)
GEOM: a Prolog geometry theorem prover
LNEC

COELHO,H.(1976b)
On a conversational interface between users and a data base
LNEC, DI Working report

COELHO,H.(1977)
Natural language and data bases
LNEC

COELHO,H.(1979a)
Aspectos do trabalho desenvolvido em 1978 pelo 'Groupe d'Intelligence Artificielle' da 'Universite' d'Aix-Marseille'-- Relatorio de uma visita de estudo realizada em Dezembro de 1978
LNEC

COELHO,H.(1979b)
TUGA's user guide
LNEC

COELHO,H.(1979c)
Notes on natural language conversations between a program and its users
LNEC

COELHO,H.(1979d)
Context and conversation:
a discussion from a semantic point of view
LNEC, DI Working report

COELHO,H.;COTTA,J.C.;PEREIRA,L.M.(1979)
How to solve it with Prolog
LNEC

COELHO,H.;COTTA,J.C.(1979)
Interaccao coloquial com o computador
LNEC

COLBY,K.M.;ENEA.H.(1967)
Heuristic methods for computer understanding of natural language in context-restricted on-line dialogues
Mathematical Biosciences 1

COLBY,K.R. et al.(1974)
Pattern matching rules for the recognition of natural language dialogue expression
Amer. Journ. of Comp. Linguistics

COLMERAUER,A.;KANOUI,H.;PASERO,R.;ROUSSEL,P.(1973)
Un systeme de communication homme-machine en francais
Univ. d'Aix-Marseille, G.I.A.

COLMERAUER,A.(1974)
Programmation en langue naturelle
Univ. d'Aix-Marseille, G.I.A.

COLMERAUER,A.(1975)
Les grammaires de metamorphose
Univ. d'Aix-Marseille, G.I.A.

COLMERAUER,A.(1977)
An interesting natural language subset
Univ. d'Aix-Marseille, G.I.A.

COLMERAUER,A;KANOUI,H.;CANEGHEM,M.van(1979)
Etude et realisation d' un systeme Prolog
Univ. d'Aix-Marseille, G.I.A.

COTTA,J.C.;SILVA,A.P.(1978)
Interaccao com bases de dados
LNEC

COULON,D.(1974)
Construction d'un modele informatique pour interpreter des enonces
Universite' Paris VI, These de docteur de troisieme cicle

DAHL,V.;SAMBUC,R.(1976)
Un systeme de banque de donnes en logique du premier ordre, en vue de sa consultation en langue naturelle
Univ. d'Aix-Marseille, G.I.A., D.E.A.

DAHL,V.(1977)
Un syteme deductif d'interrogation de banques de donnes en espagnol
Univ. d'Aix-Marseille, These de docteur de troisieme cicle

DARLINGTON,J.L.(1965)
Machine methods for proving logical arguments expressed in English
Machine Translation 8

DAVIS,R.(1976)
Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases
SRI, AIM - 283

DELL'ORCO,P. et al.(1977)
Catering for the experience and the naive user: a data base system with natural language query facilities
Proceed. of the IIASA workshop on Natural Language for Interaction with Data Bases

DOYLE,J.(1978)
A glimpse of truth maintenance
MIT, AI lab, AI Memo 461

ELLIOTT,R.W.(1965)
A model for a fact retrieval system
Univ. of Texas, Ph.D. Thesis

EMDEN,M.H. van(1975)
Programming with resolution logic
Univ. of Waterloo, Department of Computer Science, Report CS-75-30

EMDEN,M.H. van(1976)
First-order predicate logic as a high-level program language
Univ. of Edinburgh, MIP-R-106

EMDEN,M.H. van(1976)
Deductive information retrieval on virtual relational data bases
Univ. of Waterloo, Research report CS-76-42

EMDEN,M.H. van(1977)
Computation and deductive information retrieval
Univ. of Waterloo, Research report CS-77-16

EMDEN,M.H. van(1978)
Relational programming illustrated by a program for the game of Mastermind
Univ. of Waterloo, Research report CS-78-48

ENGELMORE,R.;NII,H.P.(1977)
A knowledge-based system for the interpretation of protein x-rays crystallographic data
Stanford University, Memo H PP-77-2

ERSHOV,A.P. et al.(1975)
RITA-An experimental man-computer system on a natural language basis
Proceed. of the 4IJCAI

FAYOLLE,B. de la(1976)
Analyse du Francais comme langage de commande dans un systeme de construction graphique
Univ. Scientifique et Medicale de Grenoble, These de docteur de troisieme cycle

FEIGENBAUM,E. et al.(1971)
On generality and problem solving: a case study using the DENDRAL program
Machine Intelligence no.5, Edinburgh Univ. Press

FEIGENBAUM,E.A.(1978)
The art of Artificial Intelligence--Themes and case studies of knowledge engineering
Proceed. of the National Computer Conference

FIKES,R.;HENDRIX,G.(1977)
A network-based knowledge representation and its natural deduction system
Proceed. of the 5IJCAI

FIRBAS,J.(1975)
On the thematic and the non-thematic section of the sentence
in "Style and Text", studies presented to N.E. Enkvist

GOLDSTEIN,I.P.;ROBERTS,R.B.(1977)
NUDGE, A knowledge-based scheduling program
MIT AI Lab, AI memo 405

GREEN,B. et al.(1961)
Baseball: An automatic question answerer
Proceed. of the Wastern Joint Computer Conference

GREEN,C.(1969)
Theorem-proving by resolution as a basis for question-answering systems
Machine Intelligence no.4, Edinburgh Univ. Press.

HARRIS,L.R.(1977)
ROBOT: a high performance natural language data base query system
Proceed. of the 5IJCAI

HART,P.E.(1975)
Progress on a computer based consultant
SRI, AI Group Technical Note 99

HART,P.E.;DUDA,R.O.(1977)
PROSPECTOR--a computer-based consultation system for mineral exploration
Artificial Intelligence Center, Technical note no.155

HASEMAN,W.D.;WHINSTON,A.B.(1975)
Problem solving approach in data management
Proceed. of the 4IJCAI

HEIDORN,G.E.(1972)
Natural language inputs to a simulation programming system
Technical Report NPS, Monterey, Calif.

HEIDORN,G.E.(1973)
An interactive simulation programming system which converses in English
Proc. Winter Simulation Conference

HENDRIX,G.G.(1977)
Human engineering for applied natural language processing
Proceed. of the 5IJCAI

HULL,R.D.(1975)
A semantics for superficial and embedded questions
in "Formal semantics of natural language", E.L. Keenan (eds.), Cambridge University Press

KEENAN,E.L.(1972)
On semantically based grammar
Linguistic Inquiry, no.3

KEENAN,E.L.(1973)
Logic and language
Daedalus, Vol.102, no.2

KEENAN,E.L.(1978)
Logical semantics and universal grammar
Theoretical Linguistics, Vol.5, no.1

KELLOG,C. et al.(1971)
The CONVERSE natural language data management system: current states and plans
Proceed. of the Symposium on Information Storage and Retrieval, ACM

KIDD,B. et al.(1977)
Using AI to fill-out your individual income tax return
Univ. of Toronto, AI memo 77-3

KNUTH,D.E.(1971)
Top-down syntax analysis
Acta Informatica, VOL.1, Fasc.2

KOWALSKI,R.(1974a)
Logic for problem solving
Univ. of Edinburgh, DCL memo no.75

KOWALSKI,R.(1974b)
Predicate logic as a programming language
IFIP74, pp.569-574, North-Holland

LANDSBERGEN,S.P.(1976)
Syntax and formal semantics of English in PHLIQA1
in "Advances in Natural Language Processing",
Univ. of Antwerp

LEHMAN,H.(1978)
Interpretation of natural language in a information system
IBM J. Research and Development, Vol.22, no.5

LEHMANN,E.(1975)
Input processing in a German language question-answering system
IIASA Conference on Artificial Intelligence: Question-Answering systems, Viena

LEHNERT,W.(1977)
The process of question answering
Yale University, Ph.D. Thesis

LENAT,D.B.(1976)
AM: An Artificial Intelligence approach to discovery in mathematics as heuristic search
Stanford University, AI Lab, Memo AIM-286

LESSER,V.R. et al.(1974)
Organization of the HEARSAYII speech understanding system
IEEE Symposium on Speech Recognition

LEVIEN,E;MARON,M.E.(1965)
Relational data file: a tool for mechanized inference execution and data retrieval
The Rand Corporation, Memo RM-4793-PR

LINDSAY,R.K.(1963)
Inferential memory as the basis of machines which understand natural language
in "Computers and Thought", Feigenbaum (ed.)

LOPES,O.(1972)
Gramatica simbolica do Portugues
Instituto Gulbenkian de Ciencia

LUKASIEWICZ,J.(1966)
Elements of mathematical logic
Pergamon Press

MACHADO-HOLSTI,M.E.(1976)
A grammar of Portuguese: a computer model of generative transformational grammar
in "Readings in Portuguese linguistics", J.Schmidt-Radefeldt(eds.), North-Holland

MALKOVSKY,M.G.(1963)
TULIPS - teachable, understanding natural language input problem solver
Proceed. of the 4IJCAI

McDERMOT,D.V.(1974)
Assimilation of new information by a natural language-understanding system
MIT, AI Lab, Memo AI TR-291

MELLISH,C.S.(1977)
An approach to the GUS travel agent problem using Prolog
Univ. of Edinburgh, DAI Working paper no.19

MINKER,J.;VANDERBRUGH,G.J.(1972)
Representations of the language recognition problem for a theorem-prover
Univ. of Maryland, TR-199

MINSKY,M.(1974)
A framework for representing knowledge
MIT, AI Lab, AI Memo 306

MONTAGUE,R.(1970)
English as a formal language
in "Formal Philosophy", Thomason (ed.), Yale Univ. Press

MYLOPOULOS,J. et al.(1975)
TORUS - A natural language understanding system for data management
Proceed. of the 4IJCAI

NASH-WEBBER,B.L.(1978)
A formal approach to discourse anaphora
BBN Report no.3761

NICOLAS,J.M.;SYRE,J.C.(1974)
Natural language question answering and automatic deduction in the system SYNTEX
Proceed. of the IFIP Congress

NII,H.P.;FEIGENBAUM,E.A.(1977)
Rule based understanding of signals
Proceed. of the Conference on Pattern - Directed Inference Systems

NILSSON,N.(1974)
Artificial Intelligence
in "Information Processing 74", North-Holland

OLESON,C.E.(1977)
EXAMINER: a system using contextual knowledge for analysis of diagnostic behavior
Proceed. of the 5IJCAI

ORNATO,M.;ZAPRI,G.P.(1976)
An application of Artificial Intelligence in information retrieval: RESEDA project for medical biographies
Proceed. of AISB

PASERO,R.(1972)
Representation du Francais en logique du premier ordre en vue de dialoguer avec un ordinateur
Universite´ d´Aix-Marseille, These de docteur de troisieme cycle

PASERO,R.(1976)
Un essai de communication sense en langue naturelle
Univ. d´Aix-Marseille, G.I.A.

PAUKER,G. et al.(1976)
Towards the simulation of clinical cognition: taking the present illness by computer
American Journal of Medicine, Vol.60, pp.981-996

PEREIRA,F.;WARREN,D.H.D.(1978)
Definite clause grammars compared with augmented transition networks
Univ. of Edinburgh, DAI, Research report no.58

PEREIRA,L.M.(1977)
Prolog, uma linguagem de programacao em logica
LNEC

PEREIRA,L.M.;PEREIRA,F.;WARREN,D.H.D.(1978)
User's guide to Decsystem-10 Prolog
LNEC

PEREIRA,L.M.;MONTEIRO,L.F.(1978)
The semantics of parallelism and co-routining in logic programming
LNEC, 1978

PEREIRA,L.M.;COELHO,H.(1979)
A logica, instrumento da comunicacao em Portugues com o computador
LNEC

PEREIRA,L.M.(1979)
Intelligent backtracking and sidetracking in horn clause programs--the theory
UNL

PETRICK,S.R.(1973)
Semantic interpretation in the REQUEST system
IBM Thomas D. Watson Research Center

PIQUE,J.F.(1978)
Interrogation en Francais d'une base de donnes relationnelle
Univ. d'Aix-Marseille, G.I.A., D.E.A.

POPPLE,H.E. et al.(1975)
DIALOG: a model of diagnostic logic for internal medicine
Proceed. of the 4IJCAI

POPPLE,H.E.(1977)
The formation of composite hypotheses in diagnostic problem solving: an exercise in synthetic reasoning
Proceed. of the 5IJCAI

RAPHAEL,B.(1964)
A computer program which understands
Proc. AFIPS, Fall Joint Computer Conference, Vol.26

REDDY,D.R.(1975)
Speech recognition
Academic Press

REITER,R.(1977)
An approach to deductive question-answering
BBN Report no.3649

RIESBECK,C.(1975)
Conceptual analysis
in "Conceptual Information Processing", Schank (ed.), North-Holland

ROBERTS,G.M.(1977)
An implementation of Prolog
Univ. of Waterloo, M.Sc Thesis

ROBINSON,J.A.(1965)
A machine-oriented logic based on the resolution principle
JACM, vol.12, pp.23-44

ROUSSEL,P.(1972)
Definition et traitement de l'egalite' formelle en demonstration automatique.
Univ. d'Aix-Marseille, G.I.A., These de docteur de troisieme cycle

ROUSSEL,P.(1975)
Prolog, manuel d'utilisation
Univ. d'Aix-Marselle, G.I.A.

SCHANK,R.C.;COLBY,K.(1973)
Computer models of thought and language
Freeman

SCHANK,R.C. et al.(1973)
MARGIE: memory, analysis, response generation and inferences on English
Proceed. of the 3IJCAI

SCHANK,R.C.;ABELSON,R.P.(1975)
Scripts, plans and knowledge
Proceed. of the 4IJCAI

SCHWARCZ,R.M. et al.(1968)
A deductive logic for answering English questions
SDC

SCRAGG,G.W.(1974)
Answering questions about processes
Univ. of California at San Diego, Ph. Thesis

SGALL,P.(1974)
Focus and contextual boundness
in "Topic and Comment, Contextual boundness and focus", Dahl (ed.)

SGALL,P.; et al.(1977)
On the role of linguistic semantics
Theoretical Linguistics, no.1/2

SHORTLIFFE,E.H.(1976)
Computer-based medical consultation: MYCIN
Elsevier

SIMMONS,R.F.(1963)
Synthetic language behavior
Data Process. Management 5, 12

SIMMONS,R.F.(1968)
Linguistic analysis of constructed responses in CAI
Univ. of Texas, TNN-86

SIMMONS,R.F.(1975)
The clowns microworld
in "Theoretical Issues in Natural Language Processing", Cambridge, Mess.

SLAGLE,J.R.(1965)
Experiments with a deductive question answering program
Comm. of the ACM, Vol.8, no.12

SOMALVICO,M.(1977)
A domain-oriented natural language understanding system for man-machine interaction with dynamic data bases
Ist. di Elect. ed Electronica del Politecnico di Milano, Relazione interna no.77-8

STOCKWELL,R.P.(1977)
Foundations of syntactic theory
Prentice-Hall

SUSSMAN,G.J.(1974)
A computational model of skill acquisition
MIT, AI lab, Ph. D. Thesis

SUSSMAN,G.J.(1977)
SLICES: at the boundary between analysis and synthesis
MIT, AI Lab, Memo 433

SUTHERLAND,W.R.;WOODS,W.A.(1974)
Natural communication with computers
BBN Report no.2976

SWARTOUT,W.R.(1977)
A DIGITALIS therapy advisor with explanations
Proceed. of the 5IJCAI

SZEREDI,P.(1978)
Prolog in Hungary
Proceed. of the Colloquium on Mathematical Logic in Programming

TARNLUND,S.-A.(1977)
Horn clause computability
Bit 17, pp.215-226

THOMPSON,F.B. et al.(1964)
DEACON breadbord summary
General Electric, RM 64 TMR 9

THOMPSON,F.B.(1966)
English for the computer
Proceed. of the Fall Joint Computer Conference

THOMPSON,F.B. et al.(1969)
REL: a rapidly extensible language system
Proce. 24th Nat'L Conf. of ACM

TRABASSO,T.(1970)
Reasoning and the processing of negation information
Invited Address, 78th Annual Convention, American Psychological Association.

TREUSCH,B.(1975)
SCRIN-Um sistema colloquiable di ritrovamento dell'informazione per biblioteche e centri di documentazione
Fondazione Dalle Molle, Working paper 16

WALKER,D.E. et al.(1975)
Speech understanding research
SRI, Annual Technical Report

WALTZ,D.(1975)
Natural language acess to a large data base: an engineering approach
Proceed. of the 4IJCAI

WALTZ,D.;GOODMAN,B.A.(1977)
Writing a natural language data base system
Proceed. of the 5IJCAI

WALTZ,D.(1978)
An English language question answering system for a large relational data base
Comm. of the ACM, VOL.21, no.7

WARREN,D.H.D.(1977a)
Implementing Prolog--compiling predicate logic programs
Univ. of Edinburgh, DAI, Research report no.39&40

WARREN,D.H.D.(1977b)
Logic programming and compiler writing
Univ. of Edinburgh, DAI, Research report no.44

WARREN,D.H.D.;PEREIRA,L.M.;PEREIRA,F.(1977c)
Prolog - the language and its implementation compared with LISP
LNEC

WEISS,S.M. et al.(1977)
A model-based consultation system for the long-term management of glaucoma
Proceed. of the 5IJCAI

WEIZENBAUM,J.(1966)
ELIZA - a computer program for the study of natural language communications between man and machine
Comm. of the ACM, Vol.9

WILENSKY,R.(1976)
Using plans to understand natural language
Proceed. of the Annual Conference of the ACM

WINOGRAD,T.(1972)
Understanding natural language
Academic Press

WOODS,W.A.(1967)
Semantics for a question-answering system
Harvard University, Computation Laboratory, Report NSF-19

WOODS,W.A.(1970)
Transition network grammars for natural language analysis
Comm. of the ACM, VOL.13, no.10

WOODS,W.A. et al.(1972)
The LUNAR sciences natural language information system: final report
BBN Report no.2378

WOODS,W.A.(1973a)
An experimental parsing system for transition network grammars
in "Natural Language Processing", Rustin (ed.), Algorithmics Press

WOODS,W.A.(1973b)
Progress in natural language understanding. An application to lunar geology
National Computer Conference

WOODS,W.A. et al.(1976a)
Speech understanding systems: final report
BBN Report no.3438

WOODS,W.A.(1976b)
Semantics and quantification in natural language question answering
BBN Report no.3687

ZIFONUN,G.(1974)
Eine konstruktsprache (KS) zur formalen reprasentation naturlich-sprachliches information
ISLIB-Info-I-3