An approach to natural language semantics in logic programming

Patrick Saint Dizier

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 389

AN APPROACH TO NATURAL LANGUAGE SEMANTICS IN LOGIC PROGRAMMING

Patrick SAINT-DIZIER

Mars 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

AN APPROACH TO NATURAL LANGUAGE SEMANTICS IN LOGIC PROGRAMMING

(UNE APPROCHE DE LA SEMANTIQUE DES LANGUES NATURELLES PAR LA PROGRAMMATION EN LOGIQUE)

Patrick SAINT-DIZIER - Mars 85

Publication Interne n° 251
34 pages

ABSTRACT :

In this paper, we present the main lines of our approach to the semantics of natural language sentences within the logic programming framework. We define tools to represent NP, VP, adjectives, determiners and adverbs with a degree of precision we think relevant for a natural language front end. The formalism used is based on first order logic augmented by PROLOG calls.

RESUME :

Nous présentons les lignes principales de notre approche de la représentation sémantique de phrases en langage naturel dans le cadre de la programmation en logique. Nous définissons plusieurs outils, destinés à des experts, pour représenter les adjectifs, les déterminants et les adverbes, avec un degré de précision que nous jugeons suffisant pour une interface en langue naturelle. Le formalisme utilisé est basé sur la logique du 1er ordre augmenté d'appels à des prédicats prédéfinis PROLOG.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

1_ INTRODUCTION

Our long term objective is the specification of a friendly man-machine interface that supports a natural language communication. Ultimately, we intend to have software that will be able to parse and to understand ordinary conversation on a limited and well defined subject. Such an interface has to be robust, helpful to the user and transportable [Sai 83]. It has also to take into account a large variety of linguistic phenomena and to deal with them in a way and with a degree of precision we think to be adequate and relevant for a natural language front end.

In this work, we use the formalisms and the tools developed within the logic programming framework as a conceptual basis. We will not insist here on the interest of using logic programming for natural language processing since recent works are sufficiently eloquent by themselves about this point [Dah 81], [McC 81], [Pal 83] and [FPe 83]. As we will show in the next sections, recent extensions of the standard predicate calculus provide tools that can be used to define a better formal model of language. Logic can be used as the underlying formalism of the parser and that of the component that computes the semantic representation of a sentence as well as the formalism of the semantic representation itself and that of the programming language. It results from this fact a greater uniformity and simplicity of expression.

Natural language analysis is used in this work for a practical goal, namely evaluating on a knowledge base queries expressed on a limited domain of a language. A knowledge base is itself a finite set of statements about a precise domain, expressed by facts, rules and integrity constraints. We think that the use use of a limited domain is not a real restriction since humans, at a given moment, process and store limited ressources.

In our work, the treatment of a sentence is functionally divided into three steps: a syntactic analysis, a semantic translation into a logical form and the evaluation of the produced formula (and eventually a certain updating of the knowledge base, but this point is out of the scope of this paper). We view the semantic representation of a sentence as the result of a rewriting process from an intermediate representation produced by the syntactic parser. Syntactic analysis can also be considered as a rewriting process and the evaluation is a rewriting process associated with simplification procedures to produce a value, a set of values or a logical formula.

In this paper, we present the main lines of our approach to the semantics of natural language sentences in logic programming. Surprisingly enough, much less attention seems to have been devoted to the semantic representation of sentences than to their syntactic analysis. Let us mention works done by [Dah 77], [Dah 79], [FPe 83], [McC 81], [McC 84] and [Fil 84]. Most of the other works, as far as we

know, use an ad'hoc representation, highly dependant on the domain. Our starting point is the formalism of the three branched quantified tree developed by [Dah 77] and [Col 79]. Our goal is to go deeper in the semantic representation of a sentence than the previous works do where some aspects of the representation remain superficial and thus inefficient and useless in a real man-machine communication. We show how entities such as determiners, negations, adjectives and some adverbs can be rewritten into a more precise representation without stating any additional restriction a priori. We develop a quite small set of tools, based on the set theory, we think to be of an adequate degree of precision for our purpose. These tools are used by human experts when they define the semantics of each lexical entity that occur in the specific domain they consider. The tools we present here are easily transportable to various domains of discourse and also to various formalisms such as those developed by [FPe 83], [McC 81], [Pal 83] and [Fil 83] within the logic programming framework.

All the tools and the formalisms we present here are directly interpretable by PROLOG. In particular, one of our main goals is to reduce higher order predicates into first order predicates augmented by PROLOG calls or to use well defined extensions to first order logic for which suitable proof procedures are available. This paper presents the target semantic representation of our system. The way it is computed is shown very breifly. Out of the scope of the present paper are, in particular, problems of non-compositionality, the specification of the interactions with the context when several representations of a sentence are possible and the cooperation between syntax and semantics. Notice that the more precise the semantic representation is, the more complex and context dependant is its computation. There is, in fact, a compromise to find between the degree of precision of the semantic representation and the degree of complexity involved by its computation.

Our approach is quite different of that of some linguistic theories such as Montague [Dow 79] or Kamp [Kam 81]. Montague's semantics is basically compositional. We think that the immediate correspondance he defines between a syntactic structure and a logical representation do not reflects the fact that the representation of a sentence is built from its syntactic tree along with the context of its utterance. It depends also on the semantic nature of structures involved in the sentence and on their mutual relations and positions. Futhermore, the formal semantics of Montague is not directly computational, and the use of higher order predicates only postpones problems instead of solving them really. In addition, truth conditional semantics is not really usefull in AI and it can only be used to deal with declarative and conditional sentences. What we need in AI is a declarative and procedural knowledge on symbols that represent objects.

The subset of the natural language we consider here are affirmative and interrogative sentences composed of the traditional NP, AP, PP and VP. VP may have any kind of complement, but each complement is considered as been related to the verb only and not to the whole

sentence. We exclude in this work infinitive sentences, the expression of time and conditionals.

In the next section, we give the main characteristics of the knowledge base used to evaluate the semantic representation of a sentence. Next, we give the definition of the formalism we use as a starting point to our work. Finally, we give the definition of the general tools used to rewrite higher order predicates into first order ones. In section 3, we give the basic semantic representation of nouns, noun complements, verbs, negation and interrogatives. Section 4 is devoted to the semantics of determiners, section 5 to the semantics of adjectives. Finally, in section 6, we present some aspects of the semantics of adverbs. Specific tools are defined in the sections where they are used.

2_ CONTEXT AND TOOLS

In this section, we first give the main characteristics of the knowledge base we use to build the semantic representation of sentences and to evaluate them. Next, we describe the input representation of the semantic component. Finally, we give the definition of the general tools we use in the next sections to describe the semantics of lexical items

2.1_ Database Theory and Logic Programming:

We use a first order language to express a database in logical form: this language is composed of constants, variables, predicates, functions, quantifiers and the logical connectors. Futhermore, this language is augmented by two PROLOG predicates: set_of(x,P(x,...),s) where s is the set of x for which P(x,...) is true, and card(s,n) where n is the cardinal of the set s. We use here set_of in its widest definition.

We use the well known notions of term, Horn clause and well formed formula (wff). The general form of a Horn clause is:
q <--- p1 ∧ p2 ∧ ... ∧ pn
The two types of Horn clauses we need are:
(1) q <---.
q(x1,x2,...,xn) is a fact if all the xi are constants, ex.:
weight(John,120).
If some xi are variables, the clause is then a rule or an integrity constraint, ex.:
nationality(x,french).
This clause means that all the people in the database are french.
(2) q <--- p1 ∧ p2 ∧ ... ∧pn
This clause is an integrity constraint or a rule (e.g. a deduction law: q is true iff p1 and p2 and....and pn are true), ex.:
grand_father(x,z) <--- father(x,y) ∧ father(y,z).

PROLOG programs generalize relational databases because they allow the definition of rules, e.g. non-ground clauses, usually expressed in a conditional form. This kind of database is called a deductive database [Llo 84].

In addition, we consider the following statements:

(1) At a given moment, the facts and the rules the database contains are finite. Thus, a special inference rule, called the closed world assumption, due to [Rei 78], states that if a ground atom A is not a logical consequence of a program, then we can infer NOT(A).

(2) PROLOG belongs to the class of special linear definite resolution (SLD) [VEm 76]. A SLD is sound and complete: there is a SLD resolution proof for any true goal. Thus, all the possible solutions for a given goal to demonstrate are enumerated. Then:
if P(x) is a goal with a single variable x and if the corresponding instanciations of x are x1, x2, ..., xn then:
(∀ x) P(x) <==> x= x1 or x2 or.....or xn. [Cla 78] and [Llo 84].
From this point, it results that there exists a mapping for non monotonic goals to first order formulas. Predicates such as TYP, COMP or COMP_PROP (see below) can be mapped into a first order formula.

(3) In the semantic representation we produce, some formulas may contain negative literals in their body. The method usually chosen to deal with negation is to augment the SLD-resolution proof procedure with the negation as failure rule [Llo 84]:
"If A is in The SLD finite failure set of a program P, then infer NOT(A)".
This rule is sound. We make the hypothesis that there exists a certain form of completeness for the negation as failure rule, even if results for general programs are still under study.

(4) All the properties are (or can be) represented by binary predicates:
weight(John,120).
size(Mary,1.70).
european(Anne,1).
(This last example is equivalent to european(Anne).)

Relations between logic and databases have been developped in detail by D. Warren [War 83]. In his work, Warren shows that relational database retrieval can be viewed as a special case of logical deduction. He also shows how queries expressed in the PROLOG subset of first order logic can be transformed into efficient PROLOG code and, in particular, he points out the importance of the order of predicates in a formula for the evaluation process.

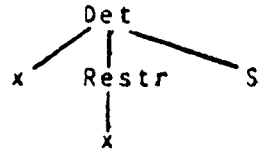
2.2_ Characteristics of the input representation:

The representation of a sentences we use as the starting point to our work is the three branched quantified tree representation defined

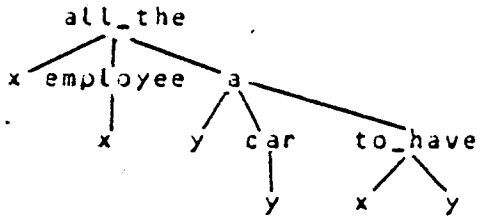
by [Dah 77] and [Col 79], but without taking into account the quantifier hierarchy rules they use. We adopt the quantifier hierarchy rules described in [Sai 85].

Very briefly, in this representation, referential words (nouns, adjectives and verbs) are translated into predicates whose arguments are typed. Determiners introduce a kind of meta-predicate whose arguments are predicates of the form:

Det(x,Restr(x),S) or:



Where Det is a determiner, Restr is the NP's translation (or a part of it) and S is the remainder of the sentence which is translated itself in the same manner. The subject NP is first translated and, thus, appears at the top of the tree, then complements in their input reading order and the verb are translated. Notice that Restr can include the representation of adjective phrases, noun complements and relative clauses. Thus, the sentence:
"All the employees have a car."
is represented by:



This representation can directly be transformed into the PROLOG subset of first order logic and thus, it is directly evaluable.

We consider here this representation as the result of the parsing process. We will show in the next sections how each node of the tree is rewritten into a more precise representation that can be a single predicate or a whole subtree. The predicates used in this representation are known to the database.

In this formalism, when a variable is created, it is typed (or restricted) by the formula Restr. Nodes of the tree can also be conjunctions (AND, OR) or negations (NOT). Proper names are translated into constants. The representation of each word is stored in a lexicon. To each word is associated at least one representation. When several representations are possible, the selection of the appropriate one depends on the semantics of other entities in the sentence or on context. For instance, the representation of a verb may depend on its subject and on some of its complements. Thus, in our system, the lexical entries of nouns, adjectives and verbs have the form of inference rules, as in [Pal 83]. Producing a semantic representation

of a particular entity in a sentence is, from a certain point of view, proving that it has been used appropriately for the domain considered. PROLOG is the ideal language for such an approach.

The input representation described here is not ambiguous. When a sentence is ambiguous, several trees are produced. During the parsing process, semantic types are used to limit the number of possible representations. However, when we compute the semantic representation of a sentence from a three branched quantified tree, we have to face additional ambiguities. Formalisms, such as [Hob 82], allow the representation of ambiguities. This approach seems to us very promising but it has to be handled with a great care because it sometimes leads to very complex representations whose evaluation is inefficient (especially when the whole sentence is involved in the ambiguity). In the next sections, we adopt a semantic representation that allow the representation of ambiguities. In the examples we present here, a semantic representation explicits all the relevant ambiguities of a sentence, but it is clear that some of them can easily be reduced by appropriate calls to the context. This is especially the case for local ambiguities. This increases considerably the efficiency of the evaluation process. Thus, our final semantic representation contains only the ambiguities that cannot directly be reduced by calls to the context.

In section 3, we give the translation of the three branched quantified tree defined above into its semantic representation for the basic components and structures: verbs, nouns, noun complements, negation and interrogatives. Specific sections are devoted to determiners (including collective and distributive readings), adjective phrases and adverbs. Notice that we do not explain how to build the semantic representation from the initial tree (this work is still under study), but we show how each node can be translated into a deeper representation. When several representations are possible, we simply explicit all of them without specifying how to select the correct one. Finally, when we define subcategories of a syntactic category (ex.: descriptive, measurable, ... for adjectives), we give some practical tools, context dependant, whose goal is to assign a precise subcategory to a given word in a given context.

2.3. General tools:

Definition 1:

A property prop is measurable if and only if, given an object x or a set of objects s, it assigns to x (or to s) a value x1, and x1 belongs to an ordered set of real or integers. It is noted:
prop(x,x1) or prop(s,x1).
We consider that the cardinal of a set (noted card(s,x1)) is a measurable property of this set.

Definition 2

For a measurable property prop some of the elements x of a set s have (e.g. prop(x,x1) is known to the knowledge base), there exists a

typical element (which differs slightly of [Woo 78], [Hob 83]), noted:
TYP(prop,x,def,val)
where:
prop is the name of the property (weight, height,...),
it is expressed by a predicate known to the knowledge base,
x is a variable that represents any object of the set s,
def is the definition of the set, often expressed by the
PROLOG call set_of(x,P,s) where P is a logical formula,
val is a real.
"val" is the average of the x1 computed from all the objects of the set s for which prop(x,x1) exists. Notice that the typical element is not itself an element of the set. When a formula contains a predicate TYP, then TYP is always true except if s is the empty set. An example of typical element is the height of french men:
TYP(height,x,set_of(x,AND(men(x),french(x)),s),val)
where val is instanciated to the average height of the french men. Only the french men whose heigth is known to the knowledge base are taken into account.

The next predicate we introduce is COMP(prop,s1,s2,F(x1,m(x2)) where:
s1, s2 are non-empty sets of objects (each set is represented by a list),
prop is a measurable property.
F(x1,m(x2)) is a function where x1 and x2 represent respectively the values, for the property prop, of any element of s1 and any element of s2, m is an arithmetical function and F is a PROLOG predicate: equal, sup or inf. The last argument of COMP can also be a conjunction of Fi(x1,mi(x2)). More formally, COMP is true iff:
$(\forall x \in s1)\ (\forall y \in s2)\ (\exists x1,x2 \in R)\ prop(x,x1) \wedge prop(y,x2) \Longrightarrow F(x1,m(x2))$
Notice that all the elements of s1 and s2 must have a value for the property prop. Notice that the notation F(x1,m(x2)) is used for more convenience. A form closer to PROLOG would be:
AND(op1(x2,m1,x3),AND(...opi(xj,mj,xj+1),F(x1,xj+1)))...)
where opi= ADD/MULT and mi is a real deduced from m
for instance:
SUP(x1,2*x2) is equivalent to:
AND(MULT(x2,2,x3),SUP(x1,x3))

The last predicate is COMP_PROP(prop1,prop2,s,F(x1,m(x2)), where:
prop1 and prop2 are two measurable properties, expressed in the same unit (ex. meters for heigth and length)
s is a non-empty set of objects
F(x1,m(x2)) has the same meaning than above.
COMP_PROP compares the values x1 and x2 of all the objects x of s for properties prop1 and prop2. COMP_PROP is true iff:
$(\forall x \in s)\ (\exists x1,x2 \in R)\ prop1(x,x1) \wedge prop2(x,x2) \Longrightarrow F(x1,m(x2))$

TYP, COMP and COMP_PROP are easily interpreted into a set of PROLOG clauses.

3_ REPRESENTATION OF NOUNS, VERBS, RELATIVE CLAUSES AND INTERROGATIVES

Nouns, verbs, relative clauses and interrogatives are represented in a way quite similar to [Dah 77] and [FPe 83]. However, we give a more precise and explicit representation of nouns, nominal compounds and verb complements. We also introduce the representation of ambiguities.

3.1 Representation of noun complements:

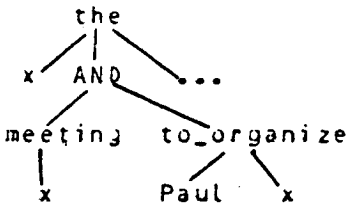
In constructions such as "The X of Y", the noun phrase "Y" is a complement or an argument [FPe 83] of "X". As explained by F. Pereira, there are two kinds of noun complements. The first type of noun complements are those which can be paraphrased by a relative clause. The other type is those that cannot. In the first case, the noun complement is represented by a subtree, that contains a verbal predicate whose subject is the modified noun (i.e. "X"). The subtree represents the relative clause the noun complement is equivalent to. The reason of this modification is that the underlying semantic relationship which exists between the two nouns is not explicit. Moreover, a large number of relationships might exist betwenn two nouns. The selection of the appropriate one depends on contextual factors. In the latter case, Y is simply represented by a binary predicate: Y_of(x,y) where x is the variable linked to X and y is the variable linked to Y. The different representations of a noun are stored in the lexicon. To each representation are associated conditions of selection that depend (1) on the semantic nature of the complemented noun and (2) on the context.

Examples:

"Paul's meeting ...."

is equivalent to: "the meeting that Paul organises ..."

it is represented by:



Such a representation makes more explicit the meaning of the noun complement. Notice also how relative clauses are translated (cf. section 3.3).

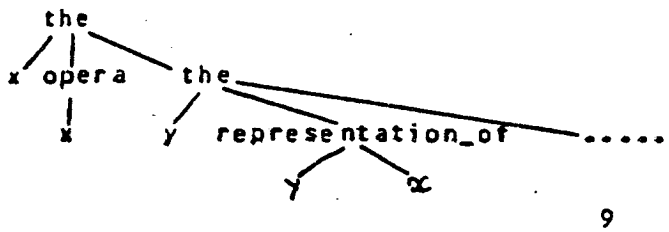
Consider now:

"The representation of the opera...."

This sentence cannot be paraphrased by:

"The representation which is of the opera ..."

It is represented by:



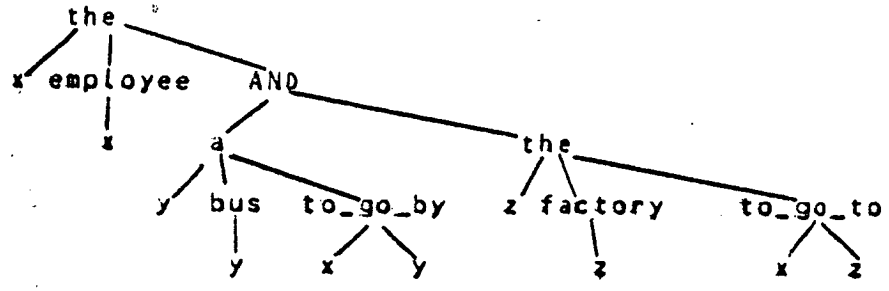
Notice that, in this case, the variable x, linked to "opera" has wider scope over y.

Nominal compounds behave in a way quite similar to noun complements, but with much more complexity. For example, consider: "a woman doctor", "a Paris flight". The last example may mean a flight bound to Paris, a flight coming from Paris or even, a flight with a stop in Paris. Here, there is a complete lack of syntactic information to guide the semantic process. It is possible to define some rules which can capture quite general forms of modifications between the modifying and the modified concept [Fin 82], but the wide variety of phenomena involved exclude, for the present moment, an automatic processing of nominal compounds.

We think that the best way is to treat them as a whole, e.g. to give to each of them a particular translation.

3.2_ Representation of verbs and verb complements:

Verbs are represented in different ways according to their complements. Intransitive verbs have only one argument that represents the subject of the verb. Transitive verbs can have two or three arguments. The representation with three arguments is used for verbs that have both a direct and an indirect object complement. For example:
"John gives Fido to Mary."
is represented by:
to_give(John,Fido,Mary).
In the other cases, verbs have two arguments. In order to have a uniform representation and to be able to deal correctly with negation, each adverbial complement is represented separately. For instance:
"The employees go by bus to the factory."
is represented by:

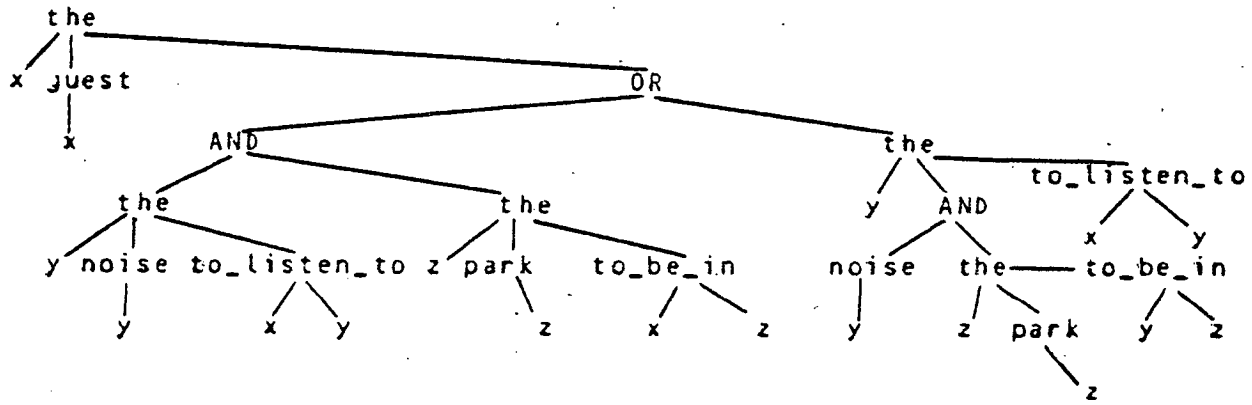


For n adverbial complements in a sentence, there are n subtrees, one for each complement.

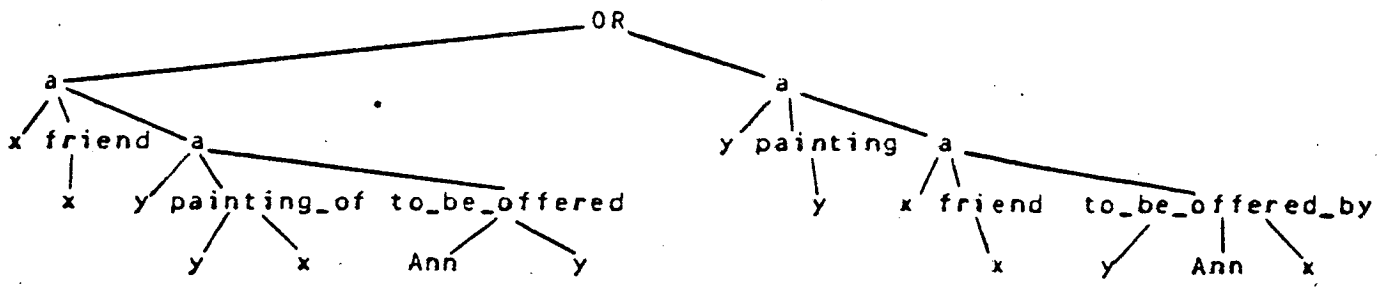
Verbs can accept sets as arguments (See examples in section 4). A set is represented by the list of its elements: x1.x2.....xn.nil. In

this case, the evaluation on the knowledge base can hold only if sets are not ordered set, i.e. a.b.c.nil is equivalent to b.c.a.nil and to a.c.b.nil etc... Thus, unification between a semantic representation and facts or rules of the knowledge base is not direct when sets are involved. The unification process is enlarged: two sets unify if and only if they have the same elements, independently of their position in the list. To unify s1 and s2, the predicate member_of(x,s2) is used for each element x of s1.

Ambiguities can appear at the level of the prepositional phrases (PP) attachment. The problem is to know where to attach a PP: to the PP it follows (it is then a noun complement) or to the verb. This problem is illustrated by sentences such as (See also [Hob 82]):
"Les invites entendaient le bruit de la fenetre."
(The guests listen to the noise of/from the window.)
"The guests listen to the noise in the park."
In this latter example, the noise or the guests can be in the park. In a way quite different of [Hob 82] because we do not use events, we explicit the ambiguities by representing each way of understanding a sentence (or a part of it) and by linking them by a "OR". Thus, the latter example is represented by:

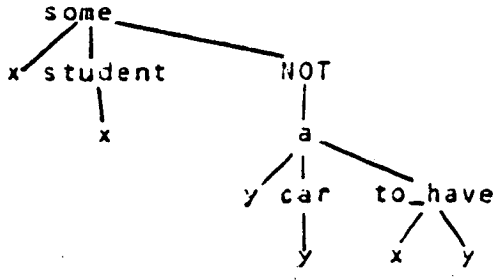


"Ann was offered a painting by a friend." is represented by:

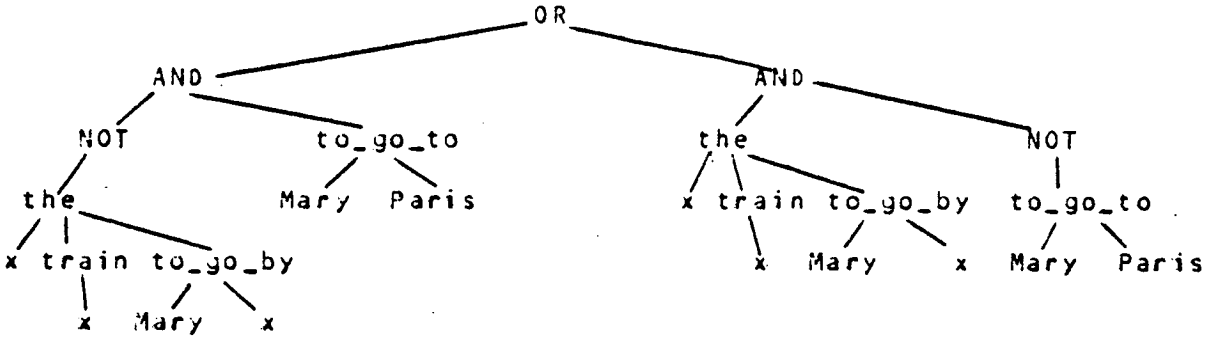


The last point to examine is negation in verb phrases. The main problem is the scope of negation. When there is only one complement in the verb phrase (VP), then the scope of the negation is the Verb or the VP except if the negation interacts with the determiner of the subject or with that of the complement (See [Sai 85]). In the sentence:

"Some students do not have a car."
The scope of the negation is the whole VP. It is represented by:



When a verb in the negative form has several complements, then the representation of the VP may be ambiguous. For instance, the sentence:
"Mary does not go by train to Paris."
may mean:
(1) that Mary takes another mean of communication than the train to go to Paris,
(2) that Mary goes by train but not to Paris.
Here again, we represent the different readings of the VP and link all the representations by a "OR". The above sentence is represented by:

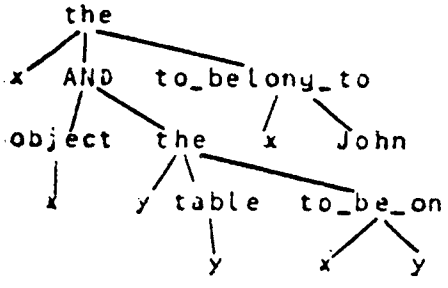


3.3_ Representation of relative clauses:

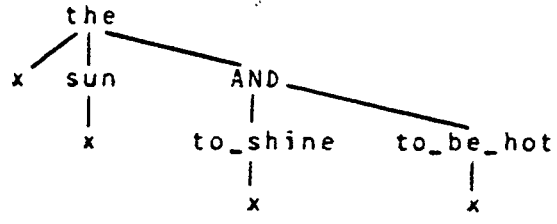
In this work, we consider only relative clauses that apply to nouns. We exclude structures such as: "I believe that ....". Futhermore, we make a distinction between relative clauses that introduce a restriction on the set of objects denoted by the noun to which they apply and relative clauses that do not introduce a restriction. This latter case is more difficult to characterize. However, it seems to me that most of them can be considered as a digression that explains some properties of the noun to which they apply. We call this class of relative clauses explicative relative clauses. Examples:
"Je l'ai rencontré qui se promenait." (I met him who was walking)
"L'homme que les autres imaginent que nous sommes." (The man that the others imagine we are)
are both descriptive relative clauses.

More formally, we say that a relative clause RC introduces a restriction on a variable x, at a given moment and in a given context, for a noun N and a determiner Det if and only if:
Denotation(Det(x) N(x) RC(..x..)) is strictly included in denotation(Det(x) N(x))

In the semantic representation, relative clauses that introduce a restriction are included in "Restr" and linked to the noun they modify by a "AND", as in [Dah 77]. Notice that our treatment of noun modifiers is uniform since adjectives are also represented in this way (cf. section 5). On the other hand, I think that an explicative relative clause has to be processed as a part of the main clause.
Examples:
"The objects that are on the table belong to John."
(the relative clause is here restrictive)
is represented by:



whereas:
"The sun which is shining is hot."
contains an explicative relative clause.
It is represented by:



3.4_ Representation of interrogatives:

Our approach of the representation of interrogative forms is similar to [FPe 81]. We distinguish five types of interrogatives according to the nature of the induced answer. In the semantic representation, each type of interrogative is represented by a different meta-predicate. A meta-predicate expresses here the nature of the subsequent evaluation process. We distinguish:
- QUEST_Y_N that represents interrogatives whose answer is yes, no, I don't know,...
- QUEST_ON that represents interrogatives whose answer is a set of

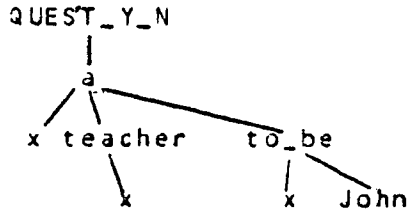
objects that satisfy the semantic representation of the sentence.

- QUEST_CARD that represents interrogatives whose answer is the cardinal of the set of objects that satisfy the semantic representation of the sentence.

- QUEST_WHY that represents interrogatives whose answer is a formula. The formula is, most of the time, the antecedent of the semantic representation of the sentence, stored in the knowledge base.

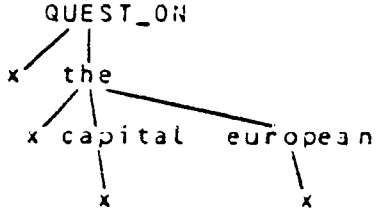
- QUEST_HOW that represents interrogatives whose answer is a formula that describes how to perform an action.

Example:
"Is John a teacher ?" is represented by:



Notice that in a formula that contains an occurence of the verb "to be", the first argument can be rewritten into the second and the subtree whose root is "to be" can be removed.

The sentence: "What are the european capitals ?" is represented by:



The first argument of QUEST_ON represents the object on which the question is. In the representation of questions that call for multiple answers, the metapredicate QUEST_ON is dominated by appropriate determiners [Sai 85].

In this section, we have described breifly formalisms to represent nouns, noun complements, verb phrases, the negation and interrogatives. We have described, in fact, a basic approach to the representation of these structures and thus our work is quite similar to [Dah 77] and [FPe 83], with, however, some extensions such as the treatment of negations, ambiguities and adverbial complements. We think that even if our approach remains quite superficial, especially for relative clauses and adverbial complements, the degree of precision we need for a classical natural language front end tends to be rather adequate.

4_ REPRESENTATION OF DETERMINERS

The role of determiners is to express the degree of determination of the noun phrase they precede. Thus, determiners introduce a quantification on the variable that represents the entity denoted by the noun they precede. Another problem is to represent the relations between the variables that appear in the semantic representation of a sentence. We have studied the problem of the range and the scope of quantified variables in [Sai 85]. We have defined a set of rewrite rules whose goal is to produce all the relevant quantifier configurations. Some rules are blocked if they contradict a fact, a rule or an integrity constraint of the knowledge base.

In this section, we examine how determiners are rewritten into a more precise semantic representation. We distinguish distributive and collective readings of a set of entities. Notice that some determiners can have different representations according to the context or to the objects that appear in the sentence. The rules we give below induced a distributive reading of a set of entities, contrary to [Dah 77] and [Col 79] but in a way quite similar to [FPe 83] and [McC 81]. Our representation is thus closer to the semantics of an evaluator implemented in PROLOG. Collective readings are detailed at the end of the section.

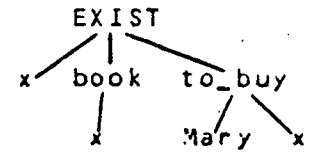
4.1_ A distributive representation of determiners:

We now describe how determiners are represented. Rules D1, D2 and D3 are similar to [FPe 83]. For each determiner Det or class of determiner, we show how the subtree whose root is this determiner is rewritten into. Notice also that the scoping problems are solved before applying the transformations decribed below.

Rule D1:

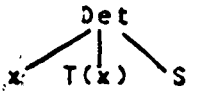
Det = { a, some, ...}

Example:
"Mary is buying a book."
is represented by:

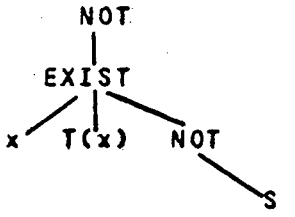


Rule D2

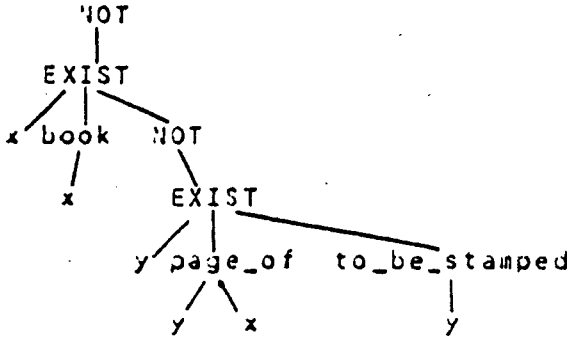
Det = { all, all the, each, every, the(plural), any, ...}



is rewritten into :

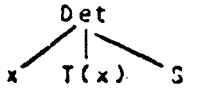


Example:
"A page of each book is stamped." is represented by:

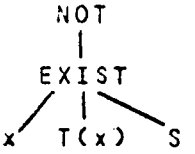


Rule D3:

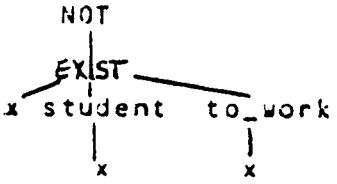
Det = { no, not any, ...}



is rewriten into:



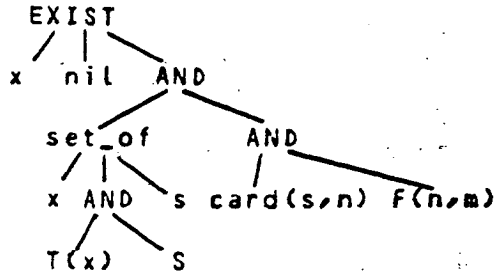
Example:
"No student works." is represented by:



Rule D4:

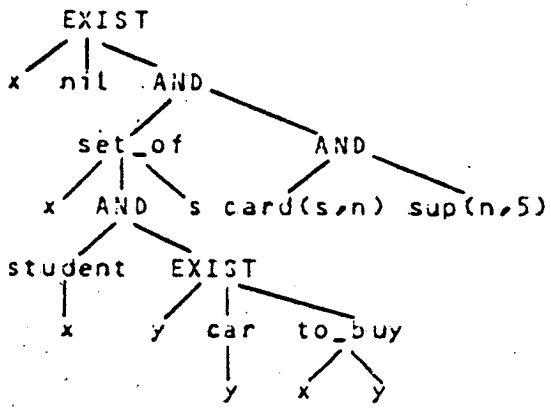
Det = {a two, three, at least four, less than three, a single, ...}

is rewritten into:



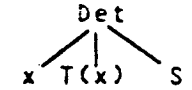
where: * m is a constant
* F is a conjunction of PROLOG calls: sup, inf, egal, diff as described in section 2.

Example:
"At least five students have bought a car."
is represented by:

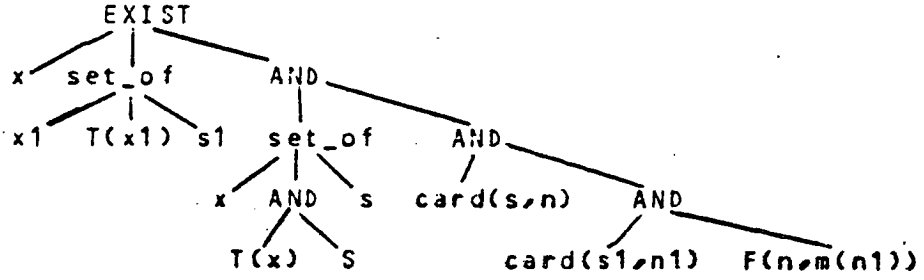


Rule D5:

Det = {several, few, a lot of, a majority, ...}

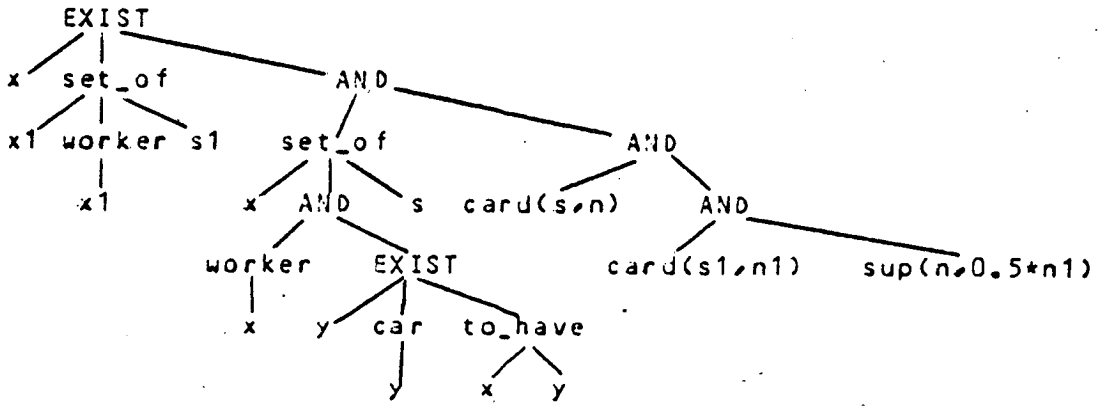


is rewritten into:



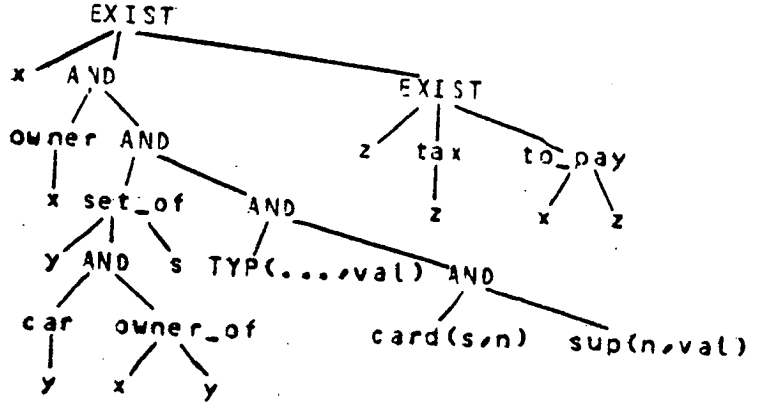
m is an arithmetical function, as defined in section 2.
s1 represents the set of entities denoted by the noun that follows Det whereas s represents the set of objects which satisfy the whole semantic representation which is in the scope of Det.

Notice that in rules D4 and D5, the second argument looses its restrictive role. The reason is not a theoretical one but only a consideration of efficiency. This presentation avoids to evaluate twice the same predicate. Another point is that it is possible, depending on the domain, to assign to m in F(n,m(n1)) a different value for each determiner. Thus, a difference of degree is made between "several", "a lot of", "many" etc...
Example:
"A lot of workers have a car."
is represented by:



We represent "a lot of" by stating that the number of people that satisfy the formula is higher that the number of people that do not.

The function F can also require the use of the typical element for the property cardinal. For instance, to represent the sentence:
"The owners of a lot of cars pay a tax."
it is necessary to compute the average number of cars per owner and then to select those who have more cars than this average number. The sentence is represented as follows:



where TYP is equal to:
TYP(card,s1,set_of(s1,AND(owner(x1),set_of(y1,AND(
car(y1),owner_of(x1,y1)),s1),s2),val)
s2 is the set of sets s1 set of cars of a given owner x1. TYP computes

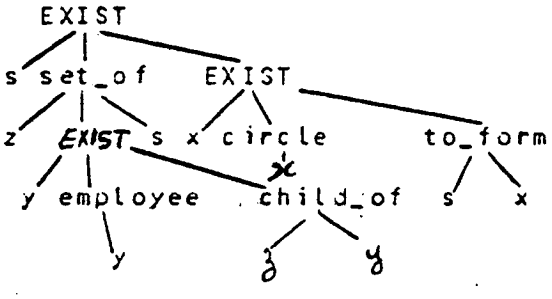
the average cardinal of the sets s1.

This latter example shows that several representations are possible for determiners such as: several, a lot of, ... In fact, the two main trends are to attach to those determiners an absolute or a relative meaning. By absolute meaning, we mean that the set of entities considered is viewed as a whole, as in "Most of the employees have a car": we consider "most of the" employees of the knowledge base. By relative meaning we mean that the set of entities we consider is structured in the form of a set of sets, as in the latter example. Relative meaning requires the use of the typical element as shown above. In both cases, the function F used remains unchanged. Our approach to relative meaning takes into account the specificities of the knowledge base with a real flexibility, because no value for a given determiner is fixed a priori.

Another form of ambiguity is suggested by J. Hobbs [Hob 82]. Consider, for instance, the sentence:
"Two examiners graded six papers."
It can be represented by the formalism defined here by representing the three following possibilities linked by an "OR":
(1) Two examiners have graded a total of 6 papers, with eventually a double correction for some papers,
(2) each examiner has graded 6 papers, and these 6 papers may be different for each examiner,
(3) each examiner has graded 6 papers and the set of the 6 papers is the same for the two examiners.

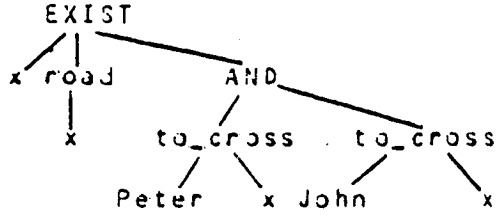
4.2 Using sets as arguments:

We now consider sentences where a collective reading is necessary. For instance, in sentences like:
"Les droites paralleles ne se coupent pas."
(parallel lines do not cross each other.)
"Les enfants des employes forment un cercle."
(The children of the employees form a circle.)
the subject argument of the verb is a set of objects. Thus, we extend the formalism described in 4.1 by allowing a variable, used as an argument, to represent a set. The second example above is represented as follows:



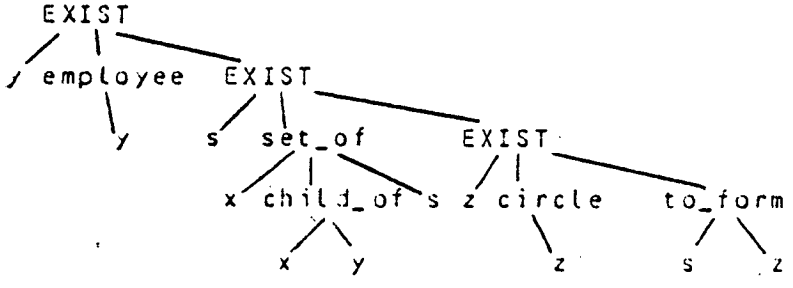
Notice that what we call a collective reading must not be confused

with sentences such as:
"Peter and John have crossed the road."
where "the road" denotes a single entity that both Peter and John have crossed separately. Peter could have crossed the road alone, whereas a child cannot form a circle alone. This sentence is represented by:



Finally, notice that a representation using sets introduces additional ambiguities in the definition of the set itself. In a sentence like:
"Les enfants de chaque employe forment un cercle."
(The children of each employee form a circle.)
A circle may by formed by:
- (a) the children of all the employees,
- (b) the children of each employee, i.e. there are as many circles as there are employees.

The first alternative is represented above, the second is represented as follows:



This representation can be extended to a NP with n complements, where there are n possible representations.

5_ REPRESENTATION OF ADJECTIVE PHRASES

We now examine the semantics of adjectives. We first give some definitions and tools specific to adjectives and then develop some examples. For general tools, see section 2.

Definition 1:
A determiner D is monotone increasing if and only if for any common noun N and for any intransitive verb phrases VI1 and VI2, such that the denotation of VI1 is a subset of the denotation of VI2 then:
D N VI1 ==> D N VI2 . [Bar 81], [Hob 83].

Determiners such as "most", "every", "some", "a", "many", "several",

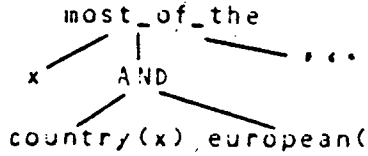
"any", "a few" are monotone increasing, but "no" and "any" are not.
For example, if:
D = most,
N = birds,
VI1 = migrate once a year,
VI2 = migrate
Since "Most birds migrate once a year" implies "Most birds migrate", the determiner "most" is monotone increasing.

Definition 2:
An adjective ADJ applied to noun N is qualificative if and only if for any monotone increasing determiner D , the set of objects denoted by D ADJ N is a subset of the set denoted by D N.

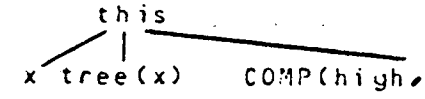
Definition 3:
Qualificative adjectives are related to measurable or descriptive properties nouns have. For measurable properties, the value of the property is a real and for descriptive properties it is a boolean.
Example:
size(John,1.75).
european(Anne,1).
Definitions 2 and 3 introduce practical tools to decide, in a given context, of the type of an adjective.

This section deals with the semantic representation of adjective phrases. We first give some properties of adjectives and then show how the different categories of adjectives can be represented. For more convenience and clarity, determiners are not rewritten into their semantic representation.

In our semantic representation, descriptive adjectives are treated as conjoined predicates [Dah 79]:
"Most of the european countries" is represented by:



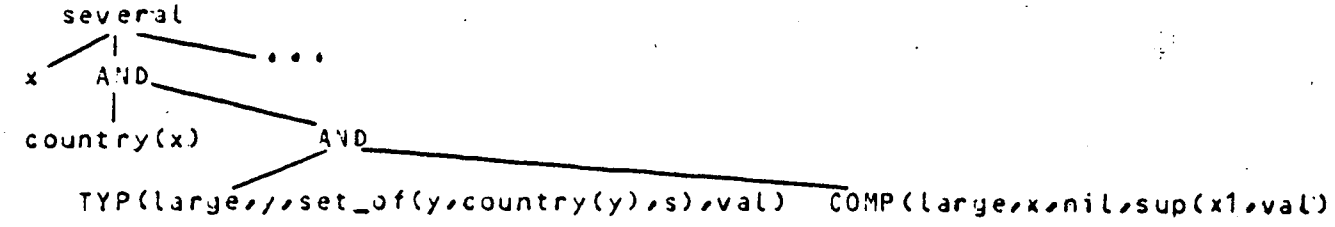
The sentence: "This tree is three meters high." is represented by:



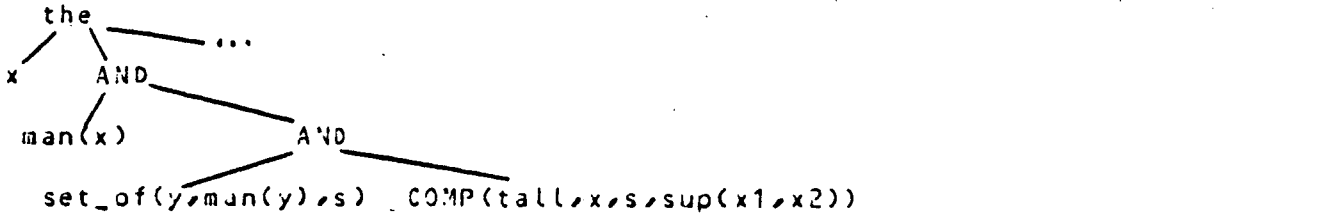
(x1 is the height of x: high(x,x1))

But, in fact, most of the adjectives appear to be implicit comparatives [Cer 83]. All adjectives based on a measurable property fall in this class. The surface NP: Det ADJ(+prop) N means that the value for the property prop for each element of the set of objects denoted by " Det Adj N " is higher than the value of the corresponding

typical element for all the objects denoted by N. Thus, a NP like: "several large countries" is represented by:



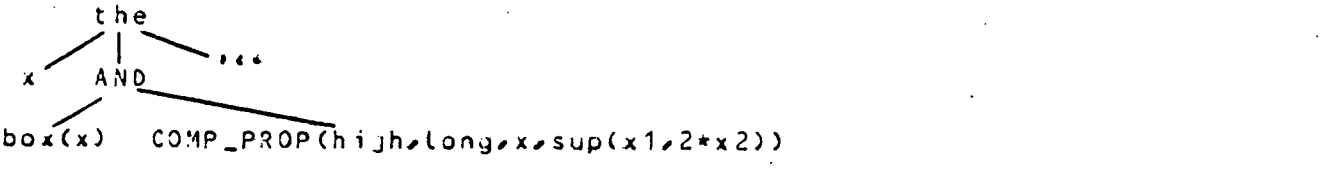
Superlatives, as in "the tallest man" are represented as follows:



In this representation, we express that x is taller than any man y. It is essential to note that in tall(x,y) we exhibit some measure y of tallness, but that does not entail that x is tall. Having tallness and being tall are distinct concepts.

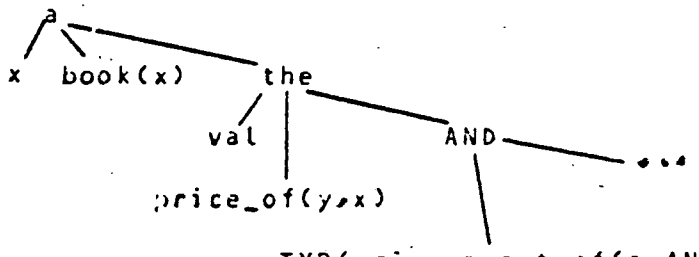
Notice that superlatives do not apply to the noun they precede but rather to the part of the NP which is in the scope of the determiner that precede the superlative [FPe 83].
In "The most famous musician that plays the violoncello"
"most famous" refers to the set of musicians that play the violoncello rather than to the most famous of all the musicians.

Comparisons between two properties can also be described by our formalism:
"the boxes two times higher than longer ..."
is represented by:



Notice that the last argument of COMP may be a conjunction of predicates:
"Paul is between 2 and 3 times richer than John." is represented by:
COMP(rich,Paul,John,AND(sup(x1,2*x2),inf(x1,3*x2)))

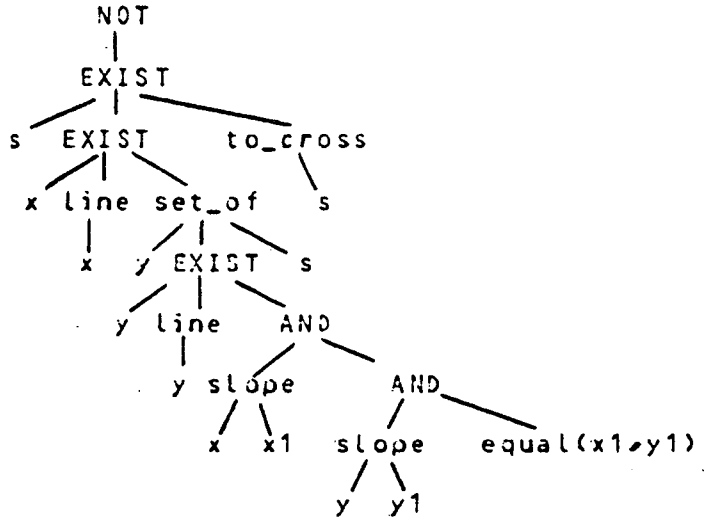
The non-qualificative adjective "average" can also be represented by our formalism. The NP "the average price of a book" is represented by:



"price" is here considered as a property of a book. In all the constructions of the form: "The average X of Y", X is, in fact, a property of Y, expressed by a noun. Other constructions, such as "the average X", where X is not directly related to a measurable property are more difficult to represent and depend most of the time on context. However, we think that they can be described by a conjunction of the predicates described here. For instance, "the average french man" can be represented by the conjunction:

(1) of the descriptive properties,
(2) of subtrees AND(TYP,COMP) of the measurable properties the average french man has.

Some non-qualificative adjectives such as "nervous", "presidential" applied to some nouns are equivalent to noun complements or to more complex structures:
"the presidential election" is equivalent to "the election of the president"
"the nervous system" is equivalent to "the system of the nerves".
"the parallel lines" is equivalent to "the lines whose slopes are equal". The evaluation of this NP implies the construction of the set of sets of parallel lines (which are here finite). For instance, the sentence:
"Parallel lines do not cross each other." is represented by:



Finally, adjectives such as "perfect", "worst", "ideal" are

superlatives that apply on a set of properties. In fact, each of these adjectives has a representation that depends on the noun they apply to. These representations are expressed in terms of a conjunction of COMP for some measurable properties and the existence or the absence of some descriptive properties.

The semantic representation of an adjective phrase is the conjunction of the partially instanciated representation of the adjectives it is composed of. These representations are stored in a lexicon. The free constituants are logical variables, they are instanciated during the semantic representation computation process. Their instanciations depend mostly on under which form the adjectives appear within the adjective phrase (superlative, comparative, modified by an adverb ...) and also on contextual information [Sai 85]. It is in particular the case for the function m in F(n,m(n1)).

6. REPRESENTATION OF ADVERBS.

Contrary to a common idea, we think that adverbs are very often used in a real man-machine communication. In particular, we think that adverbs that express an idea of quantity (quickly, well, early, ...) or an idea of intensity (very, enough, about, at least, much more, ...) have to be taken into account in any natural language front end. In this work, we will concentrate on these two types of adverbs. In order to go forward and to simplify the problem, we will consider that:

- adverbs of intensity apply to adjectives and to adverbs of quantity. They are used to refine, reinforce or weaken the meaning of the structure to which they apply.
- adverbs of quantity apply to verbs. The use of an adverb of quantity is a way to restrict (or to impose constraints on) the meaning of a verb according to one or more of its characteristics without using explicit complements. For instance, we can say:

"John gets up early." instead of "John gets up at 6 a.m."

From that point of view, we can say that adverbs operate on verbs in a way similar to adjectives on nouns. However, and contrary to adjectives, adverbs introduce additional scoping ambiguities.

It is commonly admitted that adverbs introduce higher order predicates that apply to one or more first order predicates. In works such as [Moo 81] and [Hob 85], events are used to represent actions. Then, adverbs introduce predicates that apply to these events. In this work, we simply show how some of the most useful adverbs of quantity and intensity in some precise contexts can be represented by the tools developed above, e.g. without the need of second order predicates. We think that the trend of transforming higher order predicates into first ones is very promising. However, and especially for adverbs, theoretical problems (and extensions to traditional PROLOG) remains to be solved. What we do here is simply to suggest an approach to deal with adverbs.

In our natural language understanding system adverbs are stored in

a lexicon with their semantic representation. This means that the meaning of an adverb may (1) differ from a subset of a language to another subset (2) depend on the semantic features of the NP in which it occurs.

In the next paragraphs, we first show how adverbs applied to adjectives can be represented. Then, we examine the representation of adverbs that apply to verbs. We make a distinction between those that are directly related to an adverbial property of the verb (e.g. which are more or less equivalent to an adverbial complement) and those that express the frequency of an event (ex. often). Finally, we show how some scoping problems can be taken into account. This latter point will lead us to introduce some extensions to the original tools presented above.

6.1_ Adverbs in adjective phrases:

The semantics of adverbs of intensity applied to adjective phrases can be expressed by two ways:
(1) by a modification of the function F(). The kind of modification they involve depends on the adverb. We distinguish three types of modifications:

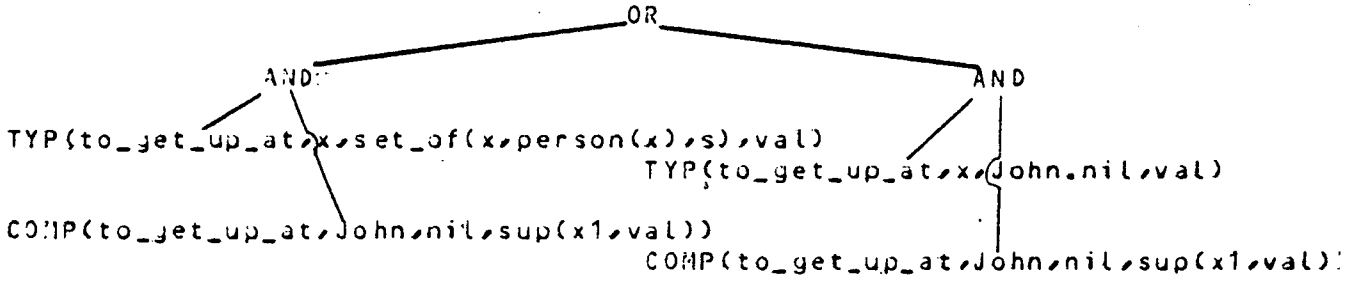
- adverbs, such as "very", that modify the value of m in F(x1,m(x2)). For example, "a large house" can be represented by stating that the number (x1) of its rooms is higher than the average number (x2) of rooms of a house: sup(x1,x2). Then, "a very large house" can be represented by stating that the number of its rooms is, for instance, two times higher than the average number of rooms of a house: sup(x1,2*x2).
- adverbs, such as "at least", that modify F. For instance, "Paul is three times richer than Mary" is represented by: COMP(rich,Paul,Mary,equal(x1,3*x2)), whereas "Paul is at least 3 times richer than Mary" is represented by: COMP(rich,Paul,Mary,sup(x1,3*x2)).
- finally, adverbs such as "enough" or "quite" imply that complementary predicates are added to F (Which itself remains unchanged). For instance, "tall men" denotes the set of men whose height is superior to the value of the typical element of the height of men, the adjunct of "enough" restricts this set by adding, for example, the constraint that these men are smaller than two times the value of the typical element.

(2) by adding subtrees: AND(TYP,COMP) or COMP to the initial tree. Adverbs such as "enough" or "very" can be represented in this way. For instance, "enough tall men" can be represented by stating that the size of these men is higher than the average size of men but inferior to the average size of the tall men. Such a construction can be applied recursively. This second alternative is less efficient (fro the evaluation) than the first one but it takes into account the specificities of the knowledge base with a greater flexibility.

6.2_ Adverbs in verb phrases:

Within our framework, only adverbs of quantity that can be directly related to complements that describe a measurable property of a verb can be taken into account. Thus, for example, if a sentence like:
"John gets up at 6 a.m. " is represented by:
to_get_up_at(John,6).
then the sentence: "John gets up early." can be treated because the adverb "early" is related to a complement (the hour) which is measurable. In this case, the semantic representation of an adverb is very similar to that of an adjective.

To represent "early", it is necessary to compute the typical element for the property "to_get_up_at" for all the humans for which the rising hour is known to the knowledge base. Then, we have to compare it to John's rising hour. There is also another way of understanding this sentence. It may mean that John gets up earlier than he is used to. In this case, the typical element is computed for the property "to_get_up_at" from the set of all John's rising hours known to the knowledge base. Thus the final representation of this ambiguous sentence is:



Notice that, according to the semantics of TYP and COMP described in section 2, when TYP(to_get_up_at,x,John.nil,val) is computed, all the facts to_get_up_at(John,_) are taken into account, whereas the value x1 taken into account in COMP is the last value known to the knowledge base.

The other class of adverbs we consider here are those which are related to the frequency of an action (ex.: often, rarely, ...). To take into account such adverbs, it is necessary to be able to count, in the knowledge base, the number of occurences of an action, eventually under some constraints (expressed by complements). For instance, consider the following independant statements:

Mary comes.
John comes by train.
John comes with Sue.

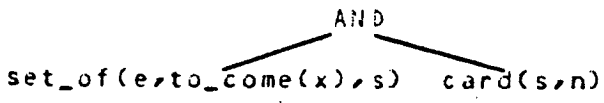
Which are represented by:

to_come(Mary).
EXIST(x, train(x),to_come_by(John,x))

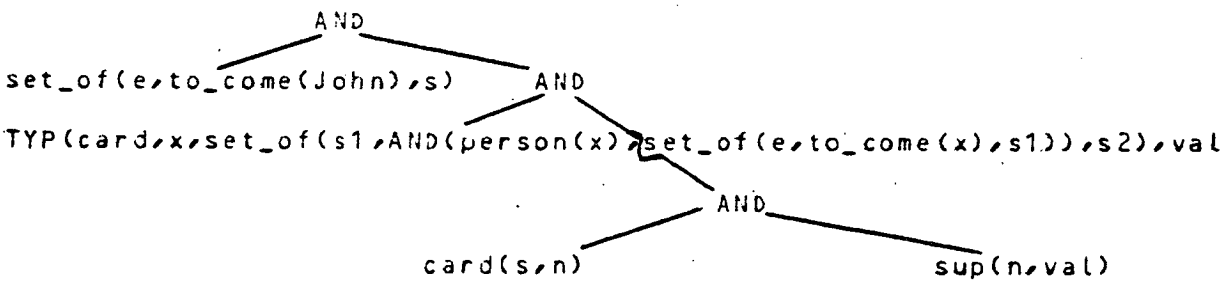
to_come_with(John,Sue).

These statements are stored in the knowledge base. From these statements it is possible to deduce that:
- 4 people have realized the action of coming,
- John has come two times,
- John has come only once by train.

In this context, the sentence "John comes often." can be represented by stating that the number of events "John comes" is higher than the average number of comings for all the other persons that have done the action of coming. The number of actions of coming for any person x is computed by:



where e is a constant. If x=John, then s= e.e.nil : a "e" is added to the list for each action of coming. The average number of actions of coming is obtained by:
TYP(card,x,set_of(s1,AND(person(x),set_of(e,to_come(x),s1)),s2),val).
s2 is a set of sets s1.
The final representation of the sentence is:



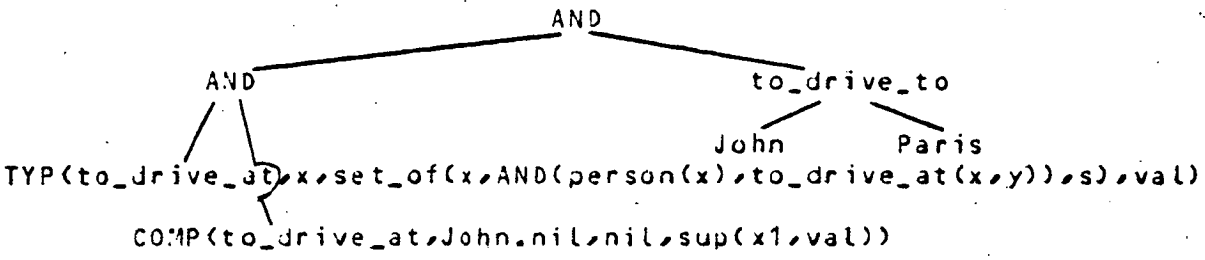
The representation of a verb phrase with complements is based on the same idea. "John come often to Paris." means that Johns comes more often to Paris than the average comings to Paris for all the other persons who come to Paris. Notice that, in our previous examples, we have not introduced any notion of time in the knowledge base.

6.3_ Scoping ambiguities:

The next point to examine is the scoping ambiguities due to complements in a VP. This difficulty has already been pointed out in [Cer 83]. Consider, for example, the two sentences:
"John is driving quickly to Paris."
"John is driving quickly on the snow."
In the first sentence the scope of "quickly" is "is driving" whereas in the second sentence it is the whole VP "is driving quickly on the snow". In addition, notice that "John is driving quickly" is ambiguous (See 6.2). For more simplicity, we will not take into account in this

paragraph this ambiguity and compare John's driving to the driving of other persons.

In the first example, the two modifiers of the verb, "quickly" and "to Paris" are represented separately, as two adverbial complements:



Where the predicate "to_drive_at" has two arguments, the fisrt represents the driver and the second, the speed:
to_drive_at(John,140).

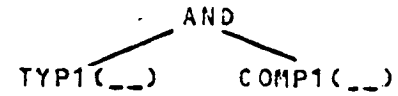
The second sentence above means that Johns drives on the snow quicker than the average speed of driving on the snow of other persons. This sntence cannot be directly represented by COMP and TYP because the first argument of these two predicates is a single predicate. What we need here is to have a full formula as the first argument of COMP and TYP. Thus, the average driving speed on the snow of a person is computed by TYP1 as follows:

```
TYP1(the(y,snow(y),the(x,person(x),AND(to_drive_on(x,y),to_drive_at(x,z)):
     z,set_of(x1,the(y1,snow(y1),the(x1,person(x1),to_drive_on(x1,y1))),s:
     val)
```

and the comparison with John's driving speed is done by using COMP1:

```
COMP1(the(y,snow(y),AND(to_drive_on(John,y),to_drive_at(John,z))),
     John.nil,nil,sup(z,val))
```

and the final representation is:



This approach of the representation of adverbs in VPs can be extented to VPs with several complements. However, with several complements, the number of ambiguities risks to increase drastically.

6.4._ Adverbs of intensity applied to adverbs of quantity:

The last point of this section is to show how adverbs of intensity can modify adverbs of quantity in VPs. In fact, adverbs of intensity applied to adverbs of quantity modify and, thus, are represented in the same way than when they apply to adjectives. Then, they are represented by a modification of the function F of the original semantic representation or by the adjunction of subtrees AND(TYP,COMP) (See 6.1).

DISCUSSION AND CONCLUSION

In this paper, we have presented the main lines of our approach to the semantics of natural language sentences in logic programming. We have defined tools, based on first order logic augmented by PROLOG calls that an expert can use to describe the semantics of each of the lexical entities involved in the domain he considers. We think that the degree of precision of these tools is relevant for a simple man-machine communication in natural language. The tools developed do not involve very complex computations and, thus, the evaluation of the semantic representation on a knowledge base defined on a limited domain is really efficient.

The main characteristics of the formalism of the tools developed and illustrated here are (1) its semantics very close to that of PROLOG and the use of a three branched quantified tree, (2) its ability to represent ambiguities, to use sets as arguments and to deal with scoping problems. We have defined a very small set of predicates (TYP, COMP, COMP_PROP), that use results of the set theory, to reduce higher order formulas into formulas for which suitable proof procedures are available. Next, we have shown up to which point our formalism is well adapted to represent the semantics of nouns, noun complements, relative clauses, verb complements, interrogatives, determiners, adjectives and adverbs. We have also given practical tools to refine the main syntactic categories.

We view the natural language understanding process as a rewriting process. The syntactic parser we use is a Gapping Grammar [Abr 84], [Dah 84]. It produces a three-branched quantified tree which is translated into the semantic representation defined here. The three branched quantified tree is built automatically by the parser, without any contextual knowledge. Each node of the tree is a word of the surface sentence, a conjunction or the negation. Then, the semantic representation of each word and contextual knowledge are used by the semantic component to build the representation described here.

The last process is the evaluation of the formula and/or the possible updatings of the knowledge base the input request implies. The updating of a knowledge base remains a very complex problem. Transformations of the input sentence are necessary, for instance to deal with existentially quantified variables. In addition a truth maintenance system is also involved. The evaluation of a request is much more simple. A system with two or three truth values [Dah 77] can be used. In addition, we have built an interpretor than translates the grammar of the semantic representation (under a Bacchus normal form) into an efficient evaluator. This interpretor is domain independant.

In the remainder of this section, we adress some of the numerous problems that remain unsolved within the framework defined here. The main problems are about the construction of the semantic representation: the cooperation between syntax and semantics, the way contextual information is used and the exact structure of the lexicon. These problems are very complex, highly interdependant, and we will simply set them here.

The first point is what we call the cooperation between syntax and semantics. Some aspects of it have been exposed in [Sai2 85]. The question is to know what syntactic information is necessary for the semantic component. Up to now, we have adopted the three-branched quantified tree as the input of our semantic component. This input representation gives an image of the original sentence with a certain degree of normalization. This normalization is due to the way the resulting tree is computed and also to the syntactic transformations done by the parsers (this is the role of several rules with gaps). To illustrate the difficulty of this problem, let us consider the case of topicalized structures. In a large majority of cases, topicalization has no impact on the semantics of a sentence. But there is at least a case where it has one. Notice the difference between these two sentences:

Every man loves a woman.
A woman every man loves.

(This example is due to J. Hobbs.)
In the first sentence, "every" has wider scope over "a" whereas in the latter, "a" has wider scope over "every". Syntax can be used to solve semantic ambiguities, to find appropriate antecedents to pronouns etc... and also to express semantic constraints in syntactic terms.

The next point is the way the semantic representation is computed. In fact, this computation is not strictly compositional, e.g. the representation of a sentence is not simply a composition of the representation of its parts. It is necessary to take into account contextual information and the fact that certain words in a sentence have a representation that depends on the semantics and the position of others. Contextual information can be used, for instance, to solve scoping ambiguities [Sai 85] or prepositional phrase attachment ambiguities. The difficulty is to be able to specify a priori the nature of the interactions with the context. Some interactions consist simply in the evaluation of a subformula or in checking if a subformula do not contradict any rule or any integrity constraint. If it is not the case, then the representation is discarded. But several interactions are much more complex to handle (see examples in [Dal 85]). The other point is words that have a representation that depends on the semantics and on the position (or role) of others. This is illustrated in [FPe 83] for the verb to have, which can have different representations depending on its subject and complements. This problem can be expressed by conditions on the environment (or by attachment rules) of an entity we want to represent. These conditions can to circumscribe the role the word we consider plays in the sentence.

The last point is the description of a lexical entity. We think that a good form for representing a lexical entry is to represent it as an inference rule, as in [Pal 83]. Thus, a semantic representation of a given word is selected if its conditions of selection are true. These conditions are a conjunction of conditions on the syntactic role of the word beeing processed and also conditions on the syntactic and the semantic roles played by other words in the sentence. If a word has several possible representations, the conditions of selection of each representation exclude each other. Both the representation of a word and the conditions of selection are domain dependant. The conditions of selection of the represntation of a word must not be confused with the conditions of selection of an overall structure for the representation of the sentence beeing parsed. This latter kind of selection rule may be, for instance, scoping rules, as in [Sai 85], or the construction of comparatives.

Two quite different approaches to the use of contextual information to build the semantic representation of a sentence were proposed by M. Palmer in [Pal 83] and M McCord in [McC 84]. In [Pal 83] calls to the context are done at the level of lexical entries. The overall structure of a sentence is built around the main verb of the sentence. In [McC 84], M. McCord describes a semantic interpretation component SEM. In SEM, there is an interleaving of several different processes: sense selection for a word, slot filling, movements of nodes (up and to the left), simplifications and exercising of semantic constraints that involve real world knowledge checkings. Finally, a formalism to describe the different ways of buiding the semantic representation of a sentence is described by H. Abramson in [Abr2 84]. The way to build the representation of a sentence is specified by one or more rules in the form of Horn clauses attached to each node of the parse tree. We think that these approaches are a good experimental and theoretical basis to develop a more complex semantic interpretor of a sentence that can build the semantics described in this paper.

REFERENCES

[Abr2 84] H. ABRAMSON Definite Clause Translation Grammars. Proc. of the intern. symposium on Logic Programming, Atlantic City, NY.

[Abr 84] V. DAHL, H. ABRAMSON On Gapping Grammars. Proc of the 2nd logic programming conf. Uppsala Univ.

[Bar 81] BARWISE J., COOPER R. Generalized quantifiers and natural language. Linguistics and philosophy, Vol 4-2.

[Cer 83] N. CERCONE A note on representing adjectives and adverbs. 5th IJCAI.

[Cla 73] K.L. CLARK Negation as failure. In Logic and databases, H. Gallaire and J. Mincker Edts, Plenum Press, NY.

[Col 79] A. COLMERAUER An Intersting subset of Natural Language. Logic Programming, CLARK and TARNLUND Edts. Academic Press.

[Dah 77] V. DAHL Un systeme deductif d'interrogation de banques de donnees en espagnol. These Universite de Marseille-Luminy. GIA.

[Dah 79] V. DAHL Quantification in a three-valued logic for natural language question-answering systems, 6th IJCAI Tokyo.

[Dah 84] V. DAHL More on Gapping Grammars. Conf. on Fifth Generation Computer systems 1984, Tokyo.

[Dal 85] A. DALADIER Programming in a Natural Language ? At what Cost ?. In "Natural language Understanding and Logic Programming" V. Dahl and P. St-Dizier Edts, North Holland Pub. Co.

[Dow 79] D.R. DOWTY Word meaning and Montague Semantics. D. Reidel Pub. Co.

[Fil 84] M. FILGUEIRAS, A. PORTO Natural Language Semantics: a Logic Programming Approach. Proc. of the 1984 Intern. Symposium on Logic Programming. Atlantic city.

[Fin 82] T.W. FININ The interpretation of Nominal Compounds in Discourse. Technical report MS CIS 1982-3 Univ. of Pennsylvania, Philadelphia.

[Gal 83] H. GALLAIRE and al. "Logic and databases: a deductive approach". Computing Survey.

[Hob 82] J. R. HOBBS Representing Ambiguity. Proc. of the 1st West Coast conf. on Formal linguistics.

[Hob 83] J.R. HOBBS An improper treatment of quantification in ordinary english. Proceedings of ACL 83.

[FPe 83] F. PEREIRA Logic for natural language analysis. SRI international technical note no 275.

[Kam 83] H. KAMP A Theory of Truth and Semantic Representation. in J. Groendijk and al. Formal methods for the study of language. Amsterdam Press

[Llo 84] J. W. LLOYD Foundations Of Logic Programming. Springer Verlag.

[McC 81] M. McCORD Using Slots and Modifiers in Logic Grammars for Natural Language Tech. report 69-80 and Artificial Intellingence 1982.

[McC 84] M. McCORD Semantic Interpretation for the EPISTLE system. Proc. of the second intern. conf. on Logic Programming , Uppasal Univ.

[Pal 83] T.W. FININ, M. PALMER Parsing with logical Variables. Conf. on

applied Natural language processing.

[Rei 78] REITER R. On reasoning by default. Artificial Intelligence 13:81.

[Sai 83] P. SAINT-DIZIER Modelling human computer interactions in a friendly man-machine interface. Proc. of the workshop on Logic Programming, Albufeira, Portugal.

[Sai 85] P. SAINT-DIZIER Handling quantifier scoping ambiguities in a semantic representation of natural language sentences. North Holland Pub. Co. in "Natural language Understanding and Logic Programming" V. DAHL and P. SAINT-DIZIER Edts.

[Sai2 85] P. SAINT-DIZIER A logic-based grammar for handling adjective phrases. Conf. on Theoretical aspects of Natural language Understanding, Halifax, Canada.

[VEm 76] Van EMDEM M.H. and KOWALSKI R.A. The semantics of predicate logic as a programming language. ACM 23.4.

[Woo 72] W. WOODS, R. KAPLAN, B. NASH-WEBBER The LUNAR sciences natural language information system. BBN report 2378.

PI 238 **Algorithme optimal de décision pour l'équivalence des grammaires simples**
Didier Caucal. 48 pages : Septembre 1984.

PI 239 **Detection and diagnosis of abrupt changes in modal characteristics of nonstationary digital signals**
Michèle Basseville, Albert Benveniste, Georges Moustakides. 26 pages : Octobre 1984.

PI 240 **Convergence optimale de l'algorithme de «réallocation-recentrage» dans le cas le plus pur**
Israël-César Lerman. 39 pages : Octobre 1984.

PI 241 **Un algorithme d'exclusion mutuelle pour une structure logique en anneau**
Michel Raynal. 12 pages : Novembre 1984.

PI 242 **Note sur l'interprétation de primitives d'action proximétriques en termes de liaisons cinématiques fictives**
Bernard Espiau. 20 pages : Novembre 1984.

PI 243 **Robust detection of signals - A large deviations approach**
Georges V. Moustakides. 16 pages : Novembre 1984.

PI 244 **Algorithmes de génération de caractères**
Gérard Hégron. 24 pages : Novembre 1984.

PI 245 **Optimal stopping times for detecting changes in distributions**
Georges V. Moustakides. 10 pages : Janvier 1985.

PI 246 **Signal : a data flow oriented language for signal processing**
Paul Le Guernic, Albert Benveniste, Patricia Bournai, Thierry Gautier. 62 pages : Janvier 1985.

PI 247 **On syntax and semantics of adjective phrases in logic programming**
Patrick Saint-Dizier. 20 pages : Février 1985.

PI 248 **Modélisation des robots manipulateurs rigides**
Michel Le Borgne. 75 pages : Février 1985.

PI 249 **Detecting changes in the A.R. parameters of a nonstationary A.R.M.A. process**
Georges V. Moustakides, Albert Benveniste. 22 pages : Mars 1985.

PI 250 **A RCMP extension to ... itiserver ...tions with concurrent classes of customers**
Jean – Yves Le Boudec. 32 pages : Mars 1985.

PI 251 **An approach to natural language semantics in logic programming**
Patrick Saint – Dizier. 34 pages : Mars 1985.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique