INRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.: 954 90 20

Rapports de Recherche

N° 381

ON SYNTAX AND SEMANTICS OF ADJECTIVE PHRASES 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

Publication Interne n° 247

Février 1985

20 pages

ON SYNTAX AND SEMANTICS OFADJECTIVE PHRASES IN LOGIC PROGRAMMING

Patrick SAINT-DIZIER
I.R.I.S.A.
Campus de Beaulieu
35042 RENNES CEDEX
FRANCE

ABSTRACT :

This work deals with the syntax and the semantics of adjective phrases within the logic programming framework. First, we make some comparisons between logic-based grammars and generalized phrase structure grammars. The criteria of comparison we consider are expressive power, extensibility, conciseness and ease of implementation. Next, we show how gaps can be expressed in an elegant way by gapping grammars, the most powerfull logic-based grammar formalism. As an example, we show how the various positions of ajdective phrases can be described by gapping grammars.

The last point of this work is devoted to the semantics of adjective phrases. First, we explain how the process of natural language understanding within the logic programming framework can be viewed as a rewriting process associated with simplification procedures. Furthermore, we point out that logic can be used as the 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. Finally, we give tools based on first-order logic augmented by PROLOG calls to represent adjective phrases with a degree of precision we think to be relevant and adequate for a natural language front end. The semantic representation produced can directly be evaluated on a knowledge base defined on a limited domain.

.../...

RESUME :

Le document traite de la syntaxe et de la sémantique des syntagmes adjectivaux dans le cadre de la programmation en logique. Nous montrons l'intérêt de l'emploi des grammaires logiques, et en particulier des gapping grammars, par rapport aux GPSG, pour décrire les différents mouvements de constituants dans une phrase. Nous donnons plusieurs exemples concernant les syntagmes adjectivaux.

Nous poursuivons en montrant l'intérêt de l'emploi du formalisme de la programmation en logique pour définir une sémantique des phrases d'une langue avec un degré de précision que nous jugeons adéquat pour une interface langue naturelle. Nous donnons, pour terminer, des outils pour définir la sémantique des syntagmes adjectivaux. Les représentations sémantiques produites sont directement évaluables sur une base de connaissances définie sur un domaine limité.

PI 234 **Architectures systoliques pour la reconnaissance de mots connectés (en anglais)**
François Charot, Patrice Frison, Patrice Quinton, 40 pages ; Août 1984.

PI 235 **A scheme of Token Tracker**
Zhao Jing Lu, 62 pages ; Septembre 1984.

PI 236 **Design of one-step and multistep adaptive algorithms for the tracking of time varying systems**
Albert Benveniste, 40 pages ; Septembre 1984.

PI 237 **The design and building of Enchère, a distributed electronic marketing system**
Jean-Pierre Banatre, Michel Bana[illegible], Guy Lapalme, Florimond Ployette, 38 pages ; Septembre 1984.

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.

ON SYNTAX AND SEMANTICS OF ADJECTIVE PHRASES IN LOGIC PROGRAMMING

Patrick SAINT-DIZIER

Publication n° 247

Février 1985

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, helpfull to the user and transportable. 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 framework for our system. 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. Futhermore, we view the natural language understanding process as a rewriting process. The surface sentence is rewritten into a syntactic representation by a set of rules. This representation is transformed by a set of rules and calls to the context into a semantic representation. Finally, the semantic representation is evaluated on a knowledge base. The evaluation process is also a rewriting process associated with simplification procedures to produce a value or a set of values.

This paper is devoted to the description of the syntax and the semantics of adjective phrases. In a sentence, adjective phrases may occupy very various positions without any significant changes in their meaning. The consequence of this matter of facts is that we need a formalism that can deal with these various positions in a quite concise and efficient way. As we explain in the next section, we think that Gapping Grammars [Abr 84], [Dah 84] are an appropriate formalism to deal with this aspect of syntax, common to many modifiers.

The second aspect of adjectives we will consider here is the nature of their semantic representation. By semantic representation computation we mean the process of mapping a syntactic tree of a well formed sentence into a certain representation of its meaning. The semantic representation is produced from an intermediate syntactic structure. Two separate levels are needed in the semantic representation computation of a sentence to deal with modifier scoping and the ambiguities they can introduce. A separate level for the semantic representation computation makes it easier to treat constructions that require syntactic manipulations (to deal with, for instance: coordinations, determiner scoping, the verb "to be ", etc...), even if it is obvious that syntax and semantics have often to work together.

In this paper we define a set of tools to build the semantic representation of adjectives within the logic programming framework. This semantic representation of a sentence can directly be evaluated on a knowledge base defined on a limited domain. We think that the use of a limited domain is not a real restriction since humans process and store limited resources.

In the next section, we introduce logic-based grammars and focus on Gapping Grammars (GG). Then, we give some properties of gaps in GG and compare GG to Generalized Phrase Structure Grammars (GPSG). Finally, we show how adjective phrases can be described by GG in a concise and elegant way. Section 2 is devoted to the semantics of adjective phrases. We first give a brief overview of the elements of the database theory we will use and then give tools to represent adjective phrases.

1_ A LOGIC GRAMMAR TO DESCRIBE THE SYNTAX OF ADJECTIVE PHRASES.

1.1_ different levels of description of a sentence:

To represent the position of any structure in a sentence, we distinguish three levels of description:

- The lower level is characterized by a quite large set of basic context free rules, written in a DCG style [PPe 80]. These rules are composed of symbols augmented by arguments whose role is to express syntactic and semantic constraints between two or more constituants in a sentence. Futhermore, PROLOG calls may appear in the right hand side of a rule. The left hand side of a rule is only composed of a non terminal symbol.

- The second level is composed of a quite small set of rules whose goal is to describe left and right structure extrapositions within a sentence: topicalizations, interrogative voice, etc... Linked to these rules are these of gaps and trace [FPe 83] [Jac 82] [Sag 82].
- The third level is a very small set of meta-rules, augmented by restrictions of application. Meta-rules express generalizations about the language generated by the grammar that are not themselves directly expressible in the latter. The input of a meta-rule is a rule of the first level described above, the output is identical to the input except in the respect of those items specifically changed by the meta-rules and possibly changes consequent upon those changes [Gaz 82]. But, in fact, the meta-rules have to be handled with a great care [Per 83]. The finite closure of the grammar is guaranteed by the fact that a meta-rule can only be applied once on a rule. Notice that meta-rules do not operate like transformational grammars since they map rules into rules whereas transformational grammars map trees into trees.

1.2_ Gapping Grammars:

We will give now a brief overview of Gapping Grammars (GG) [Abr 84], [Dah 84]. Gapping Grammars are a generalization of the well known Metamorphosis Grammars (MG) [Col 78], Definite Clause Grammars (DCG) [FPe 80] and Extraposition Grammars (XG) [FPe 83]. A GG rule allows one to "indicate where intermediate, unspecified substrings can be skipped, left unanalysed during one part of the parse and possibly reordered by the rule's application for later analysis by other rules" [Abr 84]. GG are defined formally in [Dah 84]. The left hand part of a GG rule is composed of a non-terminal symbol followed by any string of non terminal and terminal symbols and gaps. PROLOG calls can also be used. The right hand side of a GG rule is a string of non terminal and terminal symbols, of gaps in any position and PROLOG calls. GG are a very powerfull formalism that can be used to describe in a very elegant and concise way complex natural language sentences as well as formal languages. A Gapping rule is of the form:

$$\alpha_1\ gap(x_1)\ \alpha_2\ gap(x_2)\ \text{----}\ gap(x_{n-1})\ \alpha_n \rightarrow \beta$$

$$\alpha_1 \in V_N\ ,\ \alpha_i\ (i>1) \in V_N \cup V_T\ ,\ x_i \in V_T^*$$

$$G = \{\ gap(x_i)\ /\ i \in [1,n]\ \}$$

$$\beta \in V_N^* \cup V_T^* \cup G^*$$

1.3_ Gaps and Gapping Grammars:

A transformational analysis of moved substructures (or constituents) of a sentence, like wh-questions or topicalized sentences, can be taken into account here because of the fact that there is a virtual symbol of an appropriate type in the position of the

extraction site. A missing constituent X has been repositioned in the sentence or simply deleted. In the first case, what is interesting to point out is that the structure of the moved constituent is kept unchanged. Thus, a traditional logic-based grammar can be extended to a grammar with gaps without needing an additional abstract derivational level. This extension is sufficient since in the grammar rules the left hand side of the rules describes the constituant in its initial position. Note that Gapping Grammars cannot be treated as a process for mapping sentences into anothers since the set of sentences produced by moving constituents is unbounded. Gapping rules has rather to be considered as higher-level rules that describe sentence structures in a very concise way and that use in a very relevant way some properties of PROLOG (non determinism on the lenght of gaps for instance). In addition, traces can be used to tie a moved constituent to its original location or to express context-sensitivity in the rules.

Some syntactic properties of the extracted constituent can change and are determined by the position (or case) of the extraction site. P. JACOBSON calls this phenomenon strong connectivity [Jac 82]. In addition, some constraints, such as ROSS' constraints [Ros 74], have also to be taken into account [FPe 83]. The expression of modifications of properties and verifications of constraints provides a way of producing well-formed sentences and only those, even if some redundancies are not eliminated (and even if well formedness in natural language is still an open problem [Dal 85]). Within the framework of Extraposition Grammars [FPe 83] and Gapping Grammars [Abr 84], [Dah 84], strong connectivity can be taken into account by representing constraints and properties by additional features. These features are arguments of the symbols of the grammar. They can be easily and powerfully manipulated because of their status of logical variables [Pal 83] and the use of unification. Logic variables allow to create structures with free constituents. These free constituents represent structures or elements that have not yet been discovered in the sentence. It is also possible to formulate constraints on these variables which can percolate in the rules. This results in a greater flexibility in building syntactic and semantic structures of a sentence being parsed. We think, in fact, that the use of logical variables and unification are some of the major strenghts of the DCG, XG, ZGC[Sab 85] and GG formalisms. In gapping Grammars, logical variables are also used to represent gaps themselves.

1.4_ Logic grammars and GPSG:

In this section, we make a brief comparison between GPSG [Gaz 82] and logic-based grammars. We think that the most relevant criteria of comparison between two formalisms for processing natural language are expressive power (or coverage), extensibility, conciseness and ease of implementation (by using, for example, interpretors). Also relevant here are the generality of the formalism (to what extend is the formalism free of linguistic assumptions ?) and how a linguistic theory is decomposed (i.e are specified and at what level,...). We will here consentrate on GPSG and the most general form of logic grammars: Gapping Grammars. Some mathematical properties of linguistic theories

such as CFL, TG, LFG and GPSG are examined in [Per 83], and their coverage for formal languages in [Jos 83].

The central notion in GPSG is node admissibility [Gaz 82]. Besides this notion, there are two other important notions:

- categories with holes, to describe unbounded dependencies, associated with derived categories and a mechanism to build derived rules. The derived rules allow the propagation of a hole. Linking rules contain a category with a hole in their right hand side.
- metarules that build rules from others.

Finally, features are correctly distributed and controlled in rules, including default assignment by four fundamental principles: the head feature convention (HFC), the control agreement principle (CAP), the foot feature principle (FFP) and the default assignment convention (DAC) [Gaz2 82]. Very interesting and important in size parsers have been developed from this formalism with some extensions: [Saq 82], [Tho 82], [Sch 82].

The formalism of GPSG, as other grammar formalisms of the lexicalist framework, is (1) dependent on some linguistic assumptions (ex. subcategorisation: the number of the rule is a feature), (2) "do not provide a way of producing all the structures interpretable as well formed sentences and only those" [Dal 85]. Derived categories raise also many problems. First, notice that the use of derived categories is not, by itself, problematic. But, the problem is that when describing a natural language, the risk is high to have a proliferation of derived categories. Anther problem is how "slash" categories are derived from the basic set of classical categories. The derivation, in Gazdar's work, is presented as a systematic process. But thei process does not reflect any linguistic reality. There is no compelling reason for doing things this way. For each derived category, we think that only a subset of the classical categories are involved and that they depend on the category beeing slashed. In addition, as pointed out by J. Bear [Bea 82], (1) there exist different types of holes: in NP subjects gaps do not act as in object NP, (2) it has never been made explicit how features of moved constituents interact with the rule-deriving machinery.

Next, the extracted node is only visible to the node that dominates the derived category. Thus, the expression of the consequences due to the movement of a constituent are limited to the subtree whose root is the node that dominates that derived category. From these facts, it results that rules with derived categories are difficult to define as soon as the size of the language becomes large. Finally, derived categories such as A/B1 B2 ...Bn and rules, such as [A/B1 B2- ... X/B1 ... Y/B2 ...] are not allowed. This limits a recursive description of moved structures in sentences.
CONTEXT-FREE.

Logic-based grammars are context-free rules where symbols are augmented by arguments and where PROLOG calls may appear within the rules. These grammars can easily be transformed into a set of PROLOG clauses since they apart of the formalism of PROLOG.

An important feature of logic-based grammar formalisms is that they are independent a priori of any linguistic assumption or theory. In fact, any linguistic theory can be described and implemented using the logic-based grammar formalisms. In the current logic-based grammars we use categories (NP, V, N,...) as names of symbols and morphosyntactic features as arguments of the symbols, but, logic-based grammars do not priviledge a priori any category or feature: Unification Grammars [Kay 85] can be implemented within the PROLOG framework.

In XG and GG, the relation between moved constituants and their original location is directly expressed in rules and are visible from any other rule. Rules describe how constituents can be moved. The constituants that do not play any role in these movements are simply ignored and represented by gaps. Gapping rules show how constituents can be repositioned, under which conditions and with which morphosyntactic modifications, including the adjunction of words that act as markers. This results in a more limited quantity of symbols, a greater clarity and conciseness of the grammar and, as a consequence, in a good possibility of extension. In GG, gaps share some properties of essential variables [Per 83], but the power of gaps is limited and controlled since gaps can stand only for finite strings of terminal symbols. Gaps are abbreviated variables and the corresponding rules without gaps are recursively enumerable. The use of gaps associated with a PROLOG implementation yields to a computationally tractable system. In addition, several gaps are allowed within the same rule.

GG allow right as well as left extraposition of constituants. Thus, each rule can be described in the most natural way. In addition, GG (and XG in a more limited extend) allow to use marker symbols whose only function is to leave traces of the extraposed constituants. Thus, a language such as $L=\{ An Bm Cn Dm / n>0, m>0 \}$ can be described [Dah 84] by GG but not by GPSG. In XG, constraints are formulated on the respective position of traces and of extraposed constituants [FPe 83]. Finally, GG can be used to describe free word order languages.

The last point is implementation. GPSG and logic-based grammars seem to be of an equivalent ease of implementation. As both have been implemented in PROLOG, it is possible to compare their efficiency. Because of the use of gaps, i.e. unspecified strings of symbols, in GG, we think that GG are less efficient than GPSG because of a greater use of backtracking they involve. (Notice, however, that efficient interpretors for GG are still under study and the preliminary works seem to be promising.) On the other hand, for a given subset of a language, GG need less rules than GPSG.

1.5 Description of adjective phrases by Gapping Grammars:

Gapping Grammars appear to be a very suitable formalism to describe the position of adjective phrases within a sentence. Without gaps, GG can represent the ordinary attributive and complement positions of adjective phrases, as in the following examples

"A very hard working programmer...."
"This line is two inches long."
Notice that we will consider as basic forms both the attributive and complement positions of adjective phrases because of the complexity of the contextual information involved in the transformation of the first form into the latter. For instance, the sentence:
"The ear splitting sound I listen to ..."
becomes:
"The sound I listen to that splits the ears ..."
Notice:

(1) the transformation process of the verb in the progressive form "splitting" into the conjugated form "splits".

(2) the syntactic transformation in number of "ear" which is rewritten into "ears". The transformation in number of the noun "ear" implies a very complex interaction with the context that we are not now able to specify.

Adjective phrases can also appear in less obvious positions. GG can repositionned them in the standard attributive or complement position by the use of gaps and via some morphological adaptations. We now develop several examples that illustrate a quite comprehensive set of adjective phrases movements. For the clarity of the presentation, we give here isolated gapping rules, that is the reason why they may seem a little simplistic and artificial. In a complete parser, they are expressed in a more structured way.

The first example illustrates the way comparatives (or superlatives) can combine with conjunctions. The sentence:
"Peter is taller than John and Mary."
means that:
"Peter is taller than John and Peter is taller than Mary."
The corresponding GG rule is:
NP-COMP Gap(X) {and} NP-COMP NP ---> NP-COMP Gap(X) {and} SN.
Notice that this rule can be applied recursively if Gap(X) contains several conjoined NP. The rightest NP is processed first because of the conjunction that is explicitly specified before it. The other conjoined NP are only separated by a comma.

The next example is the right extraposition of a relative clause. This phenomena is not particular to relative clauses composed of an auxiliary and an adjective, but it remains, nevertheless, relevant to mention it here:
"The man that was absent is here."
becomes:
"The man is here that was absent."
The GG rule is:
Det N Rel Gap(X) ---> Det N Gap(X) Rel.

Adjectives can be concerned by topicalization. Consider, for instance, the sentence:
"He is more found of string quartets than anybody."
that becomes:
"Found of string quartets, he is more than anybody."
The same phenomena exists in french:
"Il est toujours plus heureux que n'importe qui."

becomes:
"Heureux, il l'est toujours plus que n'importe qui.". Notice here the adjunction of the pronoun "l'" which is a trace for "heureux". In fact, topicalization is a reinforcement of the meaning of the adjective in the sentence. The expanded version of the previous example is:
"Heureux, il (l') est plus heureux que n'importe qui."
For the example in french, the corresponding GG rule is:
NP V Gap(X) AP [que] ---> AP [,] NP Pro(le) V Gap(X) [que].
(We do not take into account "n'importe qui" which plays no significant role here.)

Our last example is the transformation of a sentence in the affirmative form into an interrogative sentence beginning by "how" + Adjective. For instance, the affirmative sentence:
"Sheila married yesterday a tall blond haired man."
becomes:
"How tall a blond haired man did Sheila marry yesterday ?"
In this example, there is a left extraposition of an adjective and a NP and a right extraposition of a VP. Notice also the introduction of the auxiliary "to do ". The corresponding GG rule is:
NP V(verb agreement) Gap(X) Det Gap(Y) Adj Gap(Z) ---> quest_mark(how) Adj Det Gap(Y) Gap(Z) AUX(to_do,+verb agreement) NP V(+infinitive form) Gap(X) [?].
The tense of the affirmative sentence verb is transfered to the auxiliary "to do". This can be easily realized via the use of syntactic features. Notice that in the case of the above example Gap(Y) is the empty string.

This latter rule is partly consistent with the contraints formulated by J. R. Ross [Ros 74]. It is consistent with the rule stating that "When adverbs of degree which occur in pre-adjectival or pre-adverbial positions are questioned, the questioned constituent "how", cannot be moved to the front of the sentence alone but only if the adjective or the adverb is moved with it". This can be explained by the fact that "how" is analyzed as deriving from an underlying NP and that the adjective is dominated by this NP. However, this example is not consistent with the A-over-A principle because only a part ("tall") of a larger adjective phrase ("tall blond haired") is moved to the front of the sentence.

Other transformations or movements of adjective phrases are better expressed by meta-rules because they can capture more general transformations. It is the case, for instance, of the transformation into the passive voice of sentences like:
"This place is now inhabited by men."
that becomes:
"Inhabited by men is now this place."
This point is out of the scope of the present paper.

2_ A SEMANTIC REPRESENTATION OF ADJECTIVE PHRASES.

Up to now, few works have been devoted to the representation of adjectives. Let us mention the works by: [Cer 83], [Dah 79], [FPe 83]

and [Woo 72]. We have used these works more or less as a starting point for our work. In this section, we first give some notions we need about the database theory within the logic programming framework and then develop tools to represent adjective phrases.

2.1 The database theory and logic programming:

In this section, we give a brief overview of the elements of the database theory [Gal 83] we will use in the next sections.

First, 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 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).

In addition, we consider the following statements:
(1) At a given moment, the facts and the rules the database contains are finite. The closed world assumption [Rei 78] states that facts not known to be true are false.
(2) PROLOG belongs to the class of special linear definite resolution (SLD) [VEm 76]. A SLD is complete: there is a SLD resolution proof for any true goal. All the possible solutions for a given goal to demonstrate are enumerated. K. Clark [Cla 78] has proved that:
(a) if an SLD proof procedure terminates without finding a proof for a given goal P without variables, then the negation of the goal is true.
(b) 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.
It results that there exists a mapping for non monotonic goals to first order formulas. Predicates such as set_of or card (or TYP, COMP or COMP_PROP below) can be mapped into a first order formula [FPe 83].

(c) 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).)

2.2 Representation of adjective phrases:

We now examine the semantics of adjectives. We first give some definitions and tools and then develop some examples.

Definition 1:
A determiner D is monotone increasing if and only if for any common noun N, 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 are practical tools to decide, in a given context, of the type of an adjective.

Definition 4:
For a measurable property all the elements x of a set s have, there exist 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,...),
x is a variable that represent the objects of the set,
def is the definition of the set, often expressed by the PROLOG call set_of(x,P,s) where P is a logical formula [FPe 83],

val is a real.

If prop is a measurable property then there exist a projection x1 according to property prop for each element of the set s:
(V x E s) ==> (E x1) prop(x,x1)
"val" is the average of the x1 for all the objects of the set s. Notice that the typical element is not itself an element of the set.

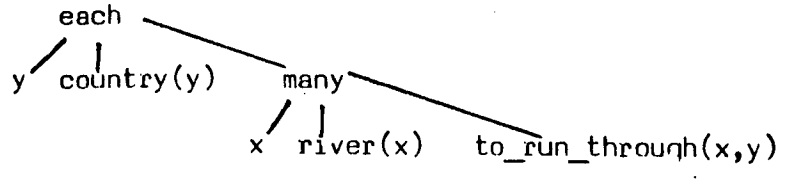
An example of typical element is the height of french men:
TYP(height,x,set_of(x,men(x) AND french(x), s),val)
where val is instanciated to the average height of the french men.

The next predicate we introduce is COMP(prop,s1,s2,F(x1,m(x2)) where:
s1, s2 are 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 represents a conjunction of PROLOG predicates: equal, sup and inf. More formally, COMP is true iff:
(Vx E s1) (Vy E s2) (Ex1,x2) prop(x,x1)$\wedge$prop(y,x2) ==> F(x1,m(x2))
During the evaluation of the logical formula that represents a sentence, F() is directly transformed into a part of a PROLOG clause.

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 height and length)
s is a set of objects
F(x1,m(x2)) has the same meaning than above.
COMP_PROP compares the values x1 and x2 of the objects x of s for properties prop1 and prop2. COMP_PROP is true iff:
(Vx E s) (Ex1,x2) prop1(x,x1)$\wedge$prop2(x,x2) ==> F(x1,m(x2))

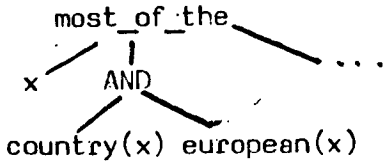
In the examples below, we use x1 and x2 with the definition given here.

We now give some examples on the way how to represent adjectives. In our formalism, a sentence is represented by a tree °Dah 79§ where:
- nodes are labeled by determiners, with three branches, conjunctions, negations or operators.
- leaves are labeled by predicates, constants or PROLOG predicates.
For instance, the sentence:
"Many rivers run through each country."
is represented by:

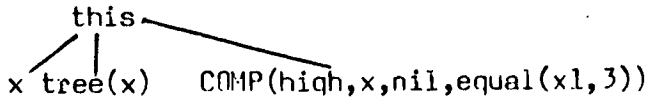


The representation of adjectives we develop below can easily be adapted to other formalisms based on logic.
In our semantic representation, descriptive adjectives are treated

as conjoined predicates [Dah 79]:
"Most of the european countries" is represented by:

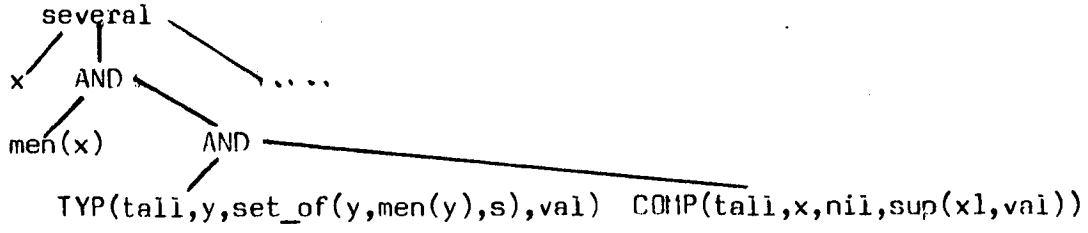


The sentence: "This tree is three meters high." is represented by:

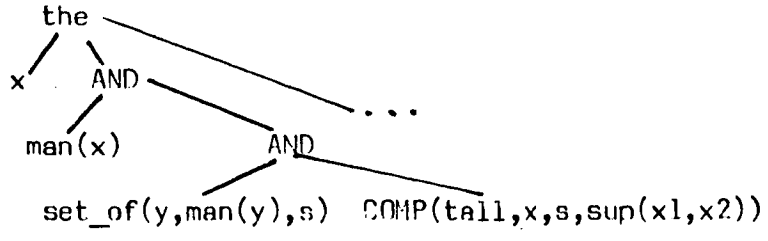


(x1 is defined by high(x,x1))

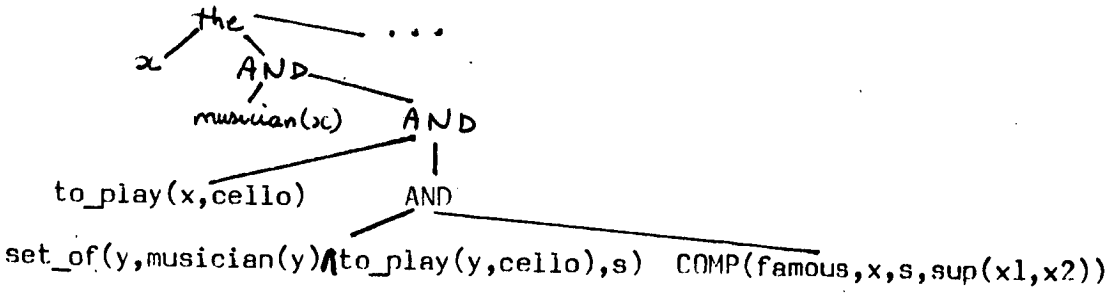
But, in fact, most of the adjectives appear to be implicit comparatives [Cer 83]. All adjectives based on a measurable property fall in this class. The surface NP: Det ADJ(+prop) N means that the value for the property prop for each element of the set of objects denoted by " Det Adj N " is higher than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



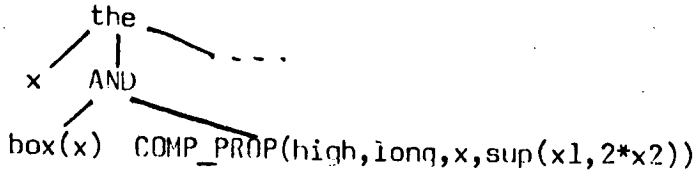
Superlatives, as in "the tallest man" are represented as follows:



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:

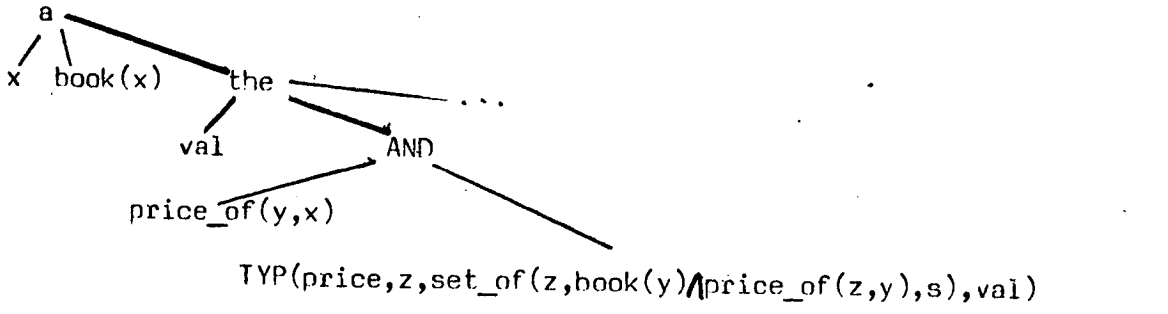


Comparatives can also be described by our formalism:
"the boxes two times higher than longer ..."
is represented by:



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,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:

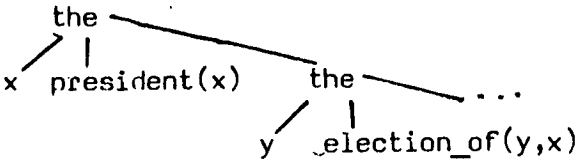


"price" is here considered as a property of a book. In all the constructions of the form: "The average X of Y", X is, in fact, a property of Y, expressed by a noun. Notice that TYP can be recursive: it can represent NP such as "the average X of the average Y of Z". Other constructions, such as "the average 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). The first example 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 depend of the noun the apply to. If we assume that to each noun are associated semantic features then we can define a specific representation of these adjectives for each semantic class. 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, i.e. they can be instanciated later in 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].

2.3 Representation of adverbs in adjective phrases:

Some adverbs (such as: very, much more, enough, at least, ...) can be used to refine, reinforce or weaken the meaning of an adjective. Most of the time, these adverbs operate as second order predicates on adjectives. In this paragraph, we will briefly show how the formalism we have developed above can be used as a practical tool to reduce these second order predicates to first order predicates augmented by PROLOG predicates.

First, notice that 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 they occur.

The semantics of the kind of adverbs mentioned above applied on adjectives can be expressed in 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)).
- 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" 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.

CONCLUSION.

In this paper, we have first pointed out the interest of using logic-based grammars, and, in particular, Gapping Grammars, to describe the various positions of moved structures within a sentence. We have then developed some examples about adjective phrases.

Next, we have described the main lines of a formalism for representing the semantics of adjective phrases within the logic programming framework with a degree of precision we think is adequate. This formalism is a practical tool an expert in linguistics can use to describe adjective phrases.

We think that our work is adaptable to various first order logical representations of natural language sentences. We will now extend this formalism to represent determiners, some adverbs and the negation and the interactions between them. Thus, we will have a uniform formalism that will be directly computational because of its syntax and semantics close to PROLOG.

REFERENCES

{Abr 84} H. ABRAMSON, V. DAHL 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.

{Bea 82} J. BEAR Gaps as Syntactic Features. Research report of the Indiana university linguistic club.

{Cer 83} N. CERCONE A note on representing adjectives and adverbs. 5th IJCAI.

{Cla 78} K.L. CLARK Negation as failure. In Logic and databases, H. Gallaire and J. Mincker Edts, Plenum Press, NY.

{Col 78} A. COLMERAUER Metamorphosis Grammars. In Bolc Edt Natural language communication with computers. Springer Verlag.

{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.

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

{Gaz 82} G. GAZDAR Phrase Structure Grammars, in "The nature of syntactic representation" Jacobson and Pullum Edts, D. Reidel Pub.Co.

{Gaz2 82} G. GAZDAR, G.K. PULLUM Generalized Phrase Structure Grammar: a Theoretical Synopsis. Research Report, Univ Of Sussex

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

{Jac 82} P. JACOBSON Evidence for gaps. In P. JACOBSON and G.K. PULLUM Edts "The nature of syntactic representation." D Reidel Pub.Co.

{Jos 83} A. JOSHI Factoring Recursion and Dependencies, 21th ACL.

{FPe 80} F. PEREIRA, D. WARREN Definite clause grammars for natural language analysis. A survey of the formalism and a comparison with ATN. Artificial intelligence no 13.

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

{Kay 85} M. KAY Unification in Grammar. In "Natural Language Understanding and Logic Programming" V. Dahl and P. St-Dizier Edts, North Holland Pub. Co.

{Pal 83} T.W. FININ, M. PALMER Parsing with logical Variables. Conf. on applied Natural language processing.

{Per 83} C.R. PERRAULT On the mathematical properties of linguistic theories. 21th ACL.

{Rei 78} REITER R. On reasoning by default. Artificial Intelligence 13:81.

{Ros 74} J.R. ROSS Excerpts from Constraints on variables in syntax. In "On N. Chomsky, critical essays" Anchor Press NY.

{Sab 85} P. SABATIER Puzzle Grammars. In "natural language understanding and logic programming" V. Dahl and P. St-Dizier Edts,

North Holland Pub. Co.

[Saq 82] I. SAG A semantic theory of "NP-movement" dependencies, in P. JACOBSON and G.K. PULLUM Edts. "The nature of syntactic representation." D Reidel Pub Co.

[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.

[Sch 82] L.K. SCHUBERT From english to logic: Context-free computation of conventional logical translation. Amer. Journ. of computational linguistics Vol 8 no 1.

[Tho 82] THOMPSON H., Chart parsing and rule schemata in GPSG. Dept of AI. Univ of Edinburgh. DAI research paper no 165.

[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.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique